Universidade de Aveiro
2007

Departamento de Electrónica, Telecomunicações e Informática

Ricardo Luís Martins de Sousa

RECEPTOR DIGITAL PARA MEDIÇÃO DE BALIZAS DE SATÉLITE

Universidade de Aveiro
2007

Departamento de Electrónica, Telecomunicações e Informática

Ricardo Luís Martins de Sousa

RECEPTOR DIGITAL PARA MEDIÇÃO DE BALIZAS DE SATÉLITE

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações (Mestrado Integrado), realizada sob a orientação científica do Dr. Armando Rocha, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente	Prof. Dr. José Fernando da Rocha Pereira professor associado da Universidade de Aveiro
orientador	Prof. Dr. Armando Rocha (Orientador) professor auxiliar da Universidade de Aveiro
vogal	Prof. Dr. Pedro Renato Tavares de Pinho professor adjunto no Instituto Superior de Engenharia de Lisboa

agradecimentos

Este trabalho assinala o fim de mais uma etapa da minha vida académica, mas não o teria conseguido concluir sem a contribuição de algumas pessoas. Por isso, não queria deixar de expressar aqui os meus mais sinceros agradecimentos a todos os que, directa ou indirectamente, contribuiram para a realização desta dissertação de mestrado.

Ao Prof. Dr. Armando Rocha, meu orientador, pela paciência e incansável disponibilidade para me ajudar a resolver os problemas mais bicudos e pela motivação que nunca deixou de me transmitir. Sem a sua enorme dedicação este trabalho não teria sido possível.

Aos meus pais e à minha irmã, que foram sempre aqueles que sofreram mais naqueles dias em que estava mais mal-humorado e impaciente, mas que mesmo assim nunca deixaram de acreditar em mim e de me apoiar sempre que foi necessário.

Ao meu amigo André, que apesar das inúmeras derrotas no PES infligidas por mim, nunca desistiu da nossa amizade e esteve sempre pronto a ouvir os meus desabafos.

Não poderia deixar de agradecer também aos meus familiares e amigos mais próximos que me aturaram nestes últimos meses, sem nunca deixarem de me apoiar.

palavras-chave

Receptor Digital, Propagação de Microondas, Medição de Balizas, Comunicações por Satélite, PLL, FLL.

resumo

A monitorização de balizas de satélite é essencial para a recolha de dados de propagação Terra-Satélite com vista à modelação dos fenómenos de atenuação, despolarização e cintilação que ocorrem na troposfera. Um receptor com um detector digital pode oferecer inúmeras vantagens tanto do ponto de vista de custos como potencialidades experimentais. Apresenta-se *software* para uma DSP que, conjuntamente com uma placa de aquisição e integrados programáveis para processamento digital de sinal, pretende ser uma alternativa a detectores analógicos. O *software* implementa um sistema de pesquisa e identificação do sinal numa determinada banda e executa uma malha de sincronismo por software. As condições de operação da malha são administradas em tempo real de forma a manter as características de seguimento e possibilitar a rápida aquisição caso o sinal se atenue de forma significativa. As componentes cartesianas de dois canais são filtradas e apresentadas a uma taxa reduzida como convém em experiências deste tipo. Apresentam-se resultados exaustivos testes de desempenho em condições muito realistas, incluindo ruído aditivo gaussiano, que atestam a solução apresentada.

keywords Digital Receiver, Microwave Propagation, Beacon Receiver, Satellite communications PLL, FLL.

abstract The monitoring of satellite beacons for propagation data experiments is mandatory to model Earth-Satellite troposphere impairments such as attenuation, depolarization and scintillation. A digital beacon detector can be an interesting solution in terms of development costs and experimental advantages. We here by describe a software package for a DSP card that, interfaced with a user developed card comprising the digitalization and digital signal processor chips, intends to improve a similar analogue implementation. The software performs the signal search procedures and executes a software tracking loop. The loop conditions are real time monitored so as loop parameters are kept constant and algorithms are implemented for fast reacquisition after deep fades occurrences. The Cartesian components are filtered to low data rate as required for propagation experiments.
A complete set of experimental arrangements for realistic testing, including additive Gaussian noise are presented as well as the results obtained are presented.

Índice

Índice de Figuras v

Acrónimos ix

Lista de Símbolos xi

1. Introdução 1

1.1. Objectivos 2

2. Malhas de Sincronização 5

2.1. PLL – *Phase Locked Loop* 6

2.1.1. Detector de Fase 7
2.1.2. Filtro de Malha 7
2.1.3. VCO 8
2.1.4. Função de Transferência 9
2.1.4.1. Filtro Activo de 2ª Ordem 9
2.1.5. Largura de Banda do Ruído 10
2.1.6. Limiar de Ruído 13
2.1.7. Seguimento Linear 13
2.1.8. Manutenção de Sincronismo (*Hold-in*) 14
2.1.9. Aquisição de Sincronismo 16
2.1.9.1. Técnicas Auxiliares à Aquisição de Sincronismo 20
2.1.9.2. Memória da Malha 20
2.1.9.3. Indicação de Sincronismo 21
2.1.9.4. Exemplos Observados de Aquisição de Sincronismo 21

2.2. Controlo Automático de Ganho (AGC) 23

2.2.1. Largura de Banda do AGC 25

2.3. FLL – *Frequency Locked Loop* 26

2.3.1. Malhas Implementadas 26
2.3.1.1. Malha com Um Pólo e Um Integrador 27
2.3.1.2. Malha com Um Zero e Duplo Integrador 29
2.3.2. Resultados Experimentais 31

3. Arquitectura do *Software* do Sistema 33

3.1. Estimação da Frequência do Sinal de Entrada 34

3.1.1. Algoritmo de Análise Espectral por FFT Complexa 36
3.1.2. Resolução Espectral 37

3.1.3. Discriminação da Risca Principal e Estimação da Densidade Espectral de Potência de Ruído ... 37
3.1.4. Incremento de Frequência (Δf) ... 41
3.1.5. Probabilidades de Sucesso ... 42
3.2. Cadeia Secundária de Filtragem e Decimação ... 42
3.3. Controlo Automático de Ganho (AGC) ... 48
3.3.1. Largura de Banda do AGC ... 48
3.3.2. Modelo de AGC ... 49
3.3.3. Simulação em *Matlab* do Modelo de AGC ... 50
3.4. Algoritmo de Administração da Malha ... 53
3.4.1. Indicação de Sincronismo ... 54
3.4.2. Estimação de CNR e "Congelamento" do NCO ... 55
3.5. Dimensionamento da PLL Digital ... 56
3.6. Envio de Dados em Tempo Real para o PC ... 57
4. Implementação do *Software* ... 59
4.1. Estrutura do Programa ... 59
4.2. Estimação da Frequência do Sinal de Entrada ... 63
4.3. Cadeia Secundária de Filtragem e Decimação ... 68
4.4. Controlo Automático de Ganho (AGC) ... 72
4.5. Indicação de Sincronismo ... 73
4.6. Estimação de CNR e "Congelamento" do NCO ... 75
4.7. PLL ... 79
4.8. FLL ... 81
4.9. Optimização de *Software* ... 83
5. Depuração de *Software* ... 87
5.1. Estimação da Frequência do Sinal de Entrada ... 87
5.2. Cadeia Secundária de Filtragem e Decimação ... 97
5.3. Controlo Automático de Ganho (AGC) ... 104
5.4. Indicação de Sincronismo ... 111
5.5. Estimação de CNR e "Congelamento" do NCO ... 112
6. Protótipo de um Receptor Digital de Dois Canais ... 115
6.1. *Hardware* ... 116

6.1.1. Sub-Sistema ADC-DRSP 117
6.1.2. Sub-Sistema DDS-DSP 118
6.1.3. Resumo dos Resultados de Avaliação da Placa 119
6.2. Migração de *Software* para o Protótipo de Dois Canais 120
6.2.1. Alterações na Configuração da EMIF 120
6.2.2. Alterações na Configuração do Porto McBSP1 121
6.2.3. Alterações na Configuração do AD6620 122
6.2.4. Alterações na Cadeia Secundária de Filtragem e Decimação 124
6.2.5. Outras Alterações 126

7. Resultados 131
7.1. Comportamento Dinâmico 131
7.2. Linearidade 136
7.3. Outras Inspecções aos Resultados 138
7.4. Resultados Obtidos com a FLL Digital 139
7.5. Resultados Obtidos com o Protótipo de Dois Canais 141

8. Conclusões e Trabalho Futuro 147

Referências Bibliográficas 151

Anexos 153
A. Estimação da Frequência do Sinal de Entrada 153
B. Cadeia Secundária de Filtragem e Decimação 154
C. PLL Digital 155
D. Análise dos Resultados dos Testes de Linearidade 156

Índice de Figuras

Figura 1.1 – Diagrama de blocos funcional do receptor digital. 2
Figura 2.1 – Diagrama de blocos de um detector coerente. 5
Figura 2.2 – Diagrama de blocos de uma PLL ou malha de seguimento de fase. 6
Figura 2.3 – Diagrama de blocos da PLL e correspondentes funções de transferência..... 9
Figura 2.4 – Filtro activo de 2ª ordem. 9
Figura 2.5 – Gráfico da largura de banda de ruído para o filtro de 2ª ordem. 12
Figura 2.6 – Diagrama de bode da PLL, com K_d = 0.5V/rad, K_O = 5KHz/V, B_L = 50Hz e ζ = 0.707. 12
Figura 2.7 – Resposta em frequência de um filtro activo de 2ª ordem. 17
Figura 2.8 – Forma de onda característica de um sinal do tipo *beat-note*. 18
Figura 2.9 – Exemplos de aquisição de sincronismo. 22
Figura 2.10 – Exemplos de aquisição de sincronismo na presença de ruído. 23
Figura 2.11 – Diagrama de blocos de um sistema de controlo automático de ganho ou AGC. 24
Figura 2.12 – Característica de um AGC típico. 25
Figura 2.13 – Diagrama de blocos de uma FLL ou malha de seguimento de frequência. 26
Figura 2.14 – Diagrama de blocos da FLL digital com um filtro de malha com um integrador. 28
Figura 2.15 – Resposta em frequência da FLL analógica e da FLL digital com um integrador. 29
Figura 2.16 – Diagrama de blocos da FLL digital com um filtro de malha com um integrador. 30
Figura 2.17 – Resposta em frequência da FLL analógica e da FLL digital com um integrador. 31
Figura 2.18 – Resultados obtidos com a FLL digital com B_L=5Hz e f_{in}=10.7MHz. 32
Figura 2.19 – Resultados obtidos com a FLL digital com B_L=10Hz e f_{in}=10.7MHz. 32
Figura 3.1 – Diagrama de blocos funcional do sistema a implementar. 33
Figura 3.2 – Fluxograma do módulo de estimação da frequência do sinal de entrada.... 35
Figura 3.3 – Varrimento da banda do sinal utilizando o NCO como oscilador de frequência variável. 36
Figura 3.4 – Espectro obtido por análise espectral por FFT. 38
Figura 3.5 – Diagrama de blocos da cadeia secundária de filtragem e decimação. 43
Figura 3.6 – Espectro do sinal ao longo da cadeia secundária de filtragem e decimação. 44
Figura 3.7 – Processamento das amostras na cadeia secundária de filtragem e decimação. 45
Figura 3.8 – Resposta em frequência do filtro FIR1. 46
Figura 3.9 – Resposta em frequência do filtro FIR2. 46

Figura 3.10 – Diagrama de blocos do AGC. 48
Figura 3.11 – Diagrama de blocos funcional do módulo de controlo automático do ganho. 49
Figura 3.12 – Função densidade de probabilidade da variável *AGC_Iout*. 50
Figura 3.13 – Modelo projectado com o *Simulink* para simulação do sistema de AGC. 51
Figura 3.14 – Componentes I e Q simuladas (em sincronismo e moduladas). 51
Figura 3.15 – Componentes I e Q filtradas. 52
Figura 3.16 – Amplitude detectada. 52
Figura 3.17 – Componentes I e Q à saída do AGC (*AGC_Iout* e *AGC_Qout*). 52
Figura 3.18 – Fluxograma do algoritmo de administração da malha. 53
Figura 3.19 – Resposta em frequência da PLL digital – *H(z)*. 57
Figura 4.1 – Fluxograma de *software* da rotina principal *main()*. 60
Figura 4.2 – Fluxograma de *software* do *loop* da DSP/BIOS e do rotina *RSI_RINT1()*. 61
Figura 4.3 – Fluxograma de *software* do módulo de estimação da frequência do sinal de entrada. 67
Figura 4.4 – Fluxograma de *software* da inicialização dos filtros da cadeia de filtragem e decimação. 70
Figura 4.5 – Fluxograma de *software* da implementação dos filtros da cadeia de filtragem e decimação. 71
Figura 4.6 – Fluxograma de *software* do módulo de controlo automático de ganho. 72
Figura 4.7 – Fluxograma de *software* do módulo de indicação de sincronismo. 74
Figura 4.8 – Fluxograma de *software* do módulo de estimação de CNR e "congelamento" do NCO. 77
Figura 4.9 – Fluxograma de *software* do "congelamento" do NCO. 79
Figura 4.10 – Fluxograma de *software* do módulo da PLL digital. 80
Figura 4.11 – Fluxograma de *software* do módulo da FLL digital. 82
Figura 5.1 – Gráficos com os espectros obtidos nos primeiros testes da função que calcula a FFT. 88
Figura 5.2 – Gerador de frequências da *Marconi Instruments* – modelo 2022. 89
Figura 5.3 – Gráficos com os espectros obtidos nos segundos testes da função que calcula a FFT. 89
Figura 5.4 – Gerador de sinal da *Hewlett Packard* – modelo 8648B. 90
Figura 5.5 – Gerador de ruído da *Hewlett Packard* – modelo HO1-3722A. 90
Figura 5.6 – Misturador RF (ZAD3 da *Minicircuits*) e *splitter* de 3 portas (ZSC3-1 da *Minicircuits*). 91
Figura 5.7 – Esquema de montagem utilizado para gerar o sinal com ruído. 91
Figura 5.8 – Analisador de espectros da *Hewlett Packard* – modelo 8593E. 92
Figura 5.9 – Espectro do ruído gerado. 92
Figura 5.10 – Espectro do sinal com o ruído adicionado. 93
Figura 5.11 – Ilustração do problema na estimação de frequência. 94
Figura 5.12 – *Leakage* num espectro do sinal de entrada calculado por FFT (Δf = 500Hz). 97

Figura 5.13 – Gráficos com os sinais à entrada e à saída do filtro FIR1 (Δf= 1.2Hz). ... 101
Figura 5.14 – Gráficos com os sinais à entrada e à saída do filtro FIR1 (Δf= 4.3Hz) 101
Figura 5.15 – Gráficos com os sinais à entrada e à saída do filtro FIR1 (Δf= 9.4Hz) 102
Figura 5.16 – Gráficos com os sinais à saída do filtro FIR2 (Δf= 1.2Hz). 102
Figura 5.17 – Gráficos com os sinais à saída do filtro FIR2 (Δf= 4.3Hz). 103
Figura 5.18 – Gráficos com os sinais à saída do filtro FIR2 (Δf= 194Hz). 103
Figura 5.19 – Gráfico com as amostras das componentes I e Q actuadas pelo AGC. 104
Figura 5.20 – Gráfico pormenorizado das amostras da componente I actuada pelo AGC. 105
Figura 5.21 – Atenuadores em cascata (*stepped attenuators*). 105
Figura 5.22 – Gráficos com a atenuação a variar entre 0 e 9dB, com incrementos de 1dB. 106
Figura 5.23 – Gráficos com a atenuação a variar entre 10 e 19dB, com incrementos de 1dB. 106
Figura 5.24 – Gráficos com a atenuação a variar entre 20 e 29dB, com incrementos de 1dB. 107
Figura 5.25 – Gráficos com a atenuação a variar entre 30 e 39dB, com incrementos de 1dB. 107
Figura 5.26 – Gráficos com a componente I actuada pelo AGC e os valores de frequência do NCO. 108
Figura 5.27 – Gerador de funções/formas de onda da *Wavetek* – modelo 180. 109
Figura 5.28 – Esquema do equipamento utilizado para modular o sinal em amplitude. 109
Figura 5.29 – Osciloscópio da *Hung Chang* – modelo 5502. 109
Figura 5.30 – Gráficos com os resultados obtidos com uma frequência modulante de 0.01Hz. 110
Figura 5.31 – Gráfico da variação da amplitude do sinal modulado. 111
Figura 5.32 – Gráficos com a indicação da malha em sincronismo. 112
Figura 5.33 – Resultados dos testes ao "congelamento" do NCO. 113
Figura 6.1 – Arquitectura do receptor digital para medição de dois canais. 115
Figura 6.2 – Vista superior da placa do receptor de dois canais. 116
Figura 6.3 – Vista lateral da placa do receptor de dois canais. 117
Figura 6.4 – Interface do subsistema ADS-DRSP com a ADC AD9328. 118
Figura 6.5 – Diagrama temporal da programação da DDS. 119
Figura 6.6 – Interface do subsistema DDS-DRSP com a DDS AD9850. 119
Figura 6.7 – Resposta em frequência do filtro RCF do AD6620. 123
Figura 6.8 – Resposta em frequência do filtro FIR1 para o receptor de dois canais. 124
Figura 6.9 – Resposta em frequência do filtro FIR2 para o receptor de dois canais. 125
Figura 7.1 – Resultados obtidos com um atenuador de 10dB e sem "congelar" o NCO. 131
Figura 7.2 – Resultados obtidos com um atenuador de 10dB e a "congelar" o NCO (f_{mod}=0.03Hz). 133

Figura 7.3 – Resultados obtidos com um atenuador de 10dB e a "congelar" o NCO (f_{mod}=0.009Hz). 134
Figura 7.4 – Componentes I e Q em detalhe durante um "congelamento" do NCO. ... 135
Figura 7.5 – Resultados obtidos com um atenuador de 20dB e a "congelar" o NCO abaixo de 27dB/Hz. 136
Figura 7.6 – Gráfico da diferença entre as atenuações medida e real em função da atenuação real. 137
Figura 7.7 – Gráfico da variância de fase em função da relação sinal-ruído. 138
Figura 7.8 – Resultados obtidos com a FLL digital sem ruído e sem modulação. 139
Figura 7.9 – Resultados obtidos com a FLL digital sem ruído e com modulação. 140
Figura 7.10 – Resultados obtidos com a FLL digital com ruído e sem modulação. 140
Figura 7.11 – Desvio de fase entre o sinal de entrada e o NCO. 141
Figura 7.12 – Resultados obtidos com um atenuador de 20dB e a "congelar" o NCO abaixo de 27dB/Hz (protótipo de dois canais). 142
Figura 7.13 – Amostras I e Q dos sinais copolar e crosspolar. 143
Figura 7.14 – Amostras I e Q dos sinais copolar e crosspolar com um atenuador de 10dB. 143
Figura 7.15 – Amostras I e Q dos sinais copolar e crosspolar com um atenuador de 20dB. 144
Figura 7.16 – Amostras I e Q dos sinais copolar e crosspolar com um cabo coaxial longo no canal crosspolar. 145
Figura 7.17 – Fase relativa entre o sinal copolar e o sinal crosspolar. 146
Figura 7.18 – *Scatter plot* das componentes I dos sinais copolar e crosspolar. 146
Figura 8.1 – Solução alternativa para a emulação de uma transferência de dados RS-232. 148
Figura 8.2 – Solução alternativa baseada na utilização de um dispositivo FPGA/CPLD. 149

Acrónimos

AGC	Controlo Automático de Ganho
ADC	Conversor Analógico-Digital
CNR	Relação Portadora-Ruído
CPLD	*Complex Programmable Logic Device*
CW	Onda Contínua
DDS	*Direct Digital Synthesizer*
DRSP	*Digital Receiver Signal Processor*
DSK	DSP *Starter Kit*
DSP	*Digital Signal Processor*
EHF	*Extremely High Frequency*
EMIF	*External Memory Interface*
FFT	*Fast Fourier Transform*
FI	Frequência Intermédia
FIR	*Finite Impulse Response*
FLL	Malha de Seguimento de Frequência
FPGA	*Field-Programmable Gate Array*
IIR	*Infinite Impulse Response*
McBSP	*Multichannel Buffered Serial Port*
NCO	Oscilador de Controlo Numérico
PD	Detector de Fase
PLL	Malha de Seguimento de Fase
RCF	RAM *Coefficient FIR Filter*
SNR	Relação Sinal-Ruído
VCO	Oscilador Controlado por Tensão

Lista de Símbolos

ω_n	Frequência natural da malha
$\overline{\theta_{ni}^2}$	Variância do ruído de fase
$\overline{\theta_{no}^2}$	Variância ou *jitter* de fase à saída da malha
Δf	Diferença inicial de frequência entre a frequência inicial do NCO e o sinal de entrada/Incremento de frequência na estimação da frequência do sinal de entrada
$\Delta\omega$	Diferença de frequência entre o sinal de entrada e o VCO/NCO
$\Delta\omega_L$	Frequência de *lock-in*
$\Delta\omega_{PI}$	Frequência de *pull-in*
$\Delta\omega_{PO}$	Frequência de *pull-out*
A	Ganho do amplificador no filtro analógico de 2ª ordem
B_L	Largura de banda de ruído
C	Condensador no filtro analógico de 2º ordem
C_1, C_2, C_3	Constantes das malhas da PLL/FLL
$F(s)$	Função de transferência do filtro de malha
f_{SAMP}	Frequência do sinal de relógio do *chip* AD6620
$G(s)$	Função de transferência em malha aberta
$H(s)$	Função de transferência em malha fechada
K_d	Ganho do detector de fase
K_o	Ganho do VCO/NCO
K_o	Ganho do detector de frequência
K_v	Ganho DC da malha aberta
M_1	Factor de decimação do primeiro filtro FIR
M_2	Factor de decimação do segundo filtro FIR
M_{TOTAL}	Factor de decimação total na cadeia secundária de filtragem e decimação
P_n	Potência de ruído
P_s	Potência do sinal de entrada
R_2, R_1	Resistências no filtro analógico de 2º ordem
$S_i(f)$	Densidade espectral de potência de ruído à entrada do filtro de malha
SNR_i	Razão sinal ruído à entrada da malha
SNR_L	Razão sinal ruído na malha
$S_o(f)$	Densidade espectral de potência de ruído à saída do filtro de malha

T	Período de amostragem
T_L	Tempo necessário para a ocorrência do *lock-in*
T_{PI}	Tempo necessário para a ocorrência do *pull-in*
v_c	Sinal de controlo à entrada do VCO/NCO
v_d	Saída do detector de fase
ζ	Constante de amortecimento
θ_d	Diferença de fase entre a saída e a entrada
θ_i	Fase do sinal de entrada
θ_o	Fase à saída do VCO/NCO
θ_v	Erro de fase estático ou erro de velocidade
ω_o	Frequência do VCO/NCO
ω_p	Frequência do pólo no filtro de malha da FLL
ω_z	Frequência do zero no filtro de malha da FLL

1. Introdução

A utilização de frequências cada vez mais mais elevadas em telecomunicações por satélite é uma realidade dados os enormes desenvolvimentos tecnológicos na eléctrónica e antenas. O satélite de telecomunicações não é agora apenas um retransmissor. Ele consegue fazer processamento *on-board* relativamente sofisticado que poderá aumentar a qualidade de serviço. Os equipamentos de recepção para o grande consumo estão também cada vez mais competitivos. As elevadas larguras de banda disponíveis (mais de 500MHz) podem oferecer serviços inovadores e a utilização da banda ka (20GHz) começa a ser explorada comercialmente.

Contudo a atmosfera introduz alguns efeitos de propagação que podem causar sérios problemas no desempenho de sistemas de telecomunicações por satélite, nomeadamente:

- Atenuação (absorção) causada por água liquida (chuva, nuvens, nevoiro) e vapor de água;
- Despolarização causada por chuva e nuvens de gelo;
- Cintilação causada por turbulência atmosférica devido à variabilidade do índice de refracção.

Todos os efeitos anteriores, excepto a atenuação por vapor de água aumentam com a frequência até pelo menos 90GHz. A implementação de uma margem de atenuação para garantir a tradicional disponibilidade de serviço de 0.01% torna-se impraticável. No entanto é possível recorrer a diversas técnicas que podem contribuir para minorar os efeitos da atenuação (*Fade Countermeasurement Methods*). A panóplia de métodos é grande e engloba o controlo de potência, a redução da taxa de transmissão, etc.

A implementação destes sistemas requer a adequada caracterização de aspectos estáticos (estatísticas cumulativas de atenuação, temperatura de ruído do céu, etc), como também de aspectos dinâmicos: taxa de variação da atenuação, duração do periodo em que o sinal está atenuado acima de um certo patamar, o tempo de retorno de atenuação, etc.

Dada a variabilidade geográfica e climática dos fenómenos meteorológicos associados à degradação do sinal no percurso troposférico, a realização de campanhas experimentais é mandatória. Estas campanhas baseiam-se na monitorização de um sinal CW (o denominado *beacon*) radiado por um satélite de preferência geo-estacionário. As caracteristicas de um *beacon* são tipicamente:

- Potência radiada equivalente entre 18 a 25dBW;
- Estabilidade diária de frequência melhor que 2kHz;

- Ruído de fase reduzido estando a potência de sinal contida numa largura de banda inferior a 50Hz;
- Variação diária de amplitude muito reduzida (menor que 0.5dB tipicamente).

A utilização de uma antena da ordem de 1.5m e um amplificador com uma figura de ruído típica de 3dB leva a que a densidade espectral da relação sinal-ruído (CNR) do sinal recebido seja, em condições de céu limpo, da ordem de 55dB/Hz. Este valor foi tomado como referência neste trabalho para vários testes.

A medição, com máxima gama dinâmica, da amplitude de sinais de baixa relação sinal-ruído (SNR) exige detecção síncrona e por conseguinte uma malha PLL/FLL.

A redução da SNR (devido à atenuação do sinal) leva a uma degradação da qualidade do sincronismo destas malhas e eventualmente à perda do mesmo. As medições usando receptores analógicos ficam comprometidas ou inviabilizadas para SNR inferior a 27dB/Hz (valor típico). A retoma da medição pode demorar algum tempo pois a reaquisição de sincronismo tem que ser auxiliada, é lenta e exige SNR superior ao limiar em que ocorreu a perda. Consequentemente poder-se-ão deixar de adquirir dados de atenuação profunda durante algum tempo.

A utilização de técnicas de processamento digital de sinal pode estender a gama dinâmica e tornar essencialmente nulo o tempo de retoma da medição se a SNR recuperar a breve trecho como é típico numa situação real. As campanhas experimentais no terreno podem ser grandemente enriquecidas já que períodos de elevada atenuação merecem muita atenção para efeitos de modelamento.

1.1. Objectivos

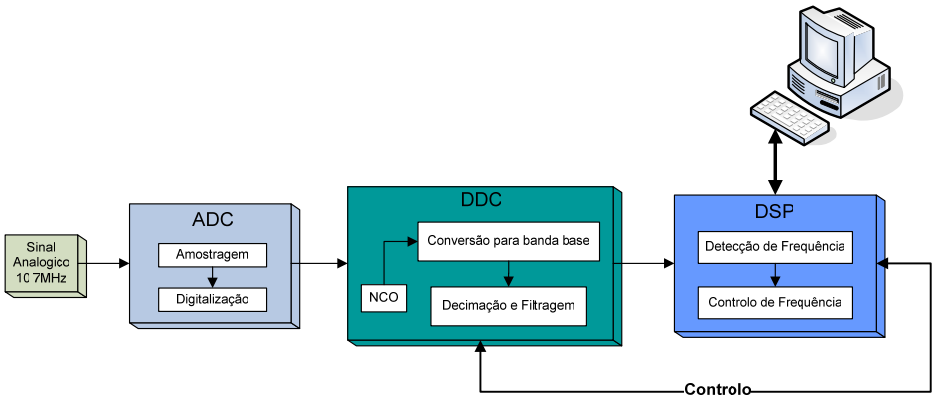

Figura 1.1 – Diagrama de blocos funcional do receptor digital.

Foram recentemente desenvolvidos com sucesso o *hardware* e *software* básico de um protótipo de receptor digital. O diagrama de blocos está apresentado na Figura 1.1 e compreende uma *card* desenvolvida para o efeito e um *kit* DSP.

Uma ADC amostra a cerca de 40MS/s uma FI de cerca de 10.7MHz. O sinal é convertido e decimado num processador digital de sinal programável. Um DSP recebe as amostras a alguns kS/s e implementa uma PLL programando em tempo real um oscilador de controlo numérico na *card*.

Posteriormente foi desenvolvida uma nova placa visando a aquisição não só do canal copolar mas também do canal crosspolar que poderá oferecer informação preciosa sobre a estrutura do canal de propagação nomeadamente informação sobre a despolarização por gelo ou mesmo a distribuição de tamanhos de gotas de chuva no trajecto.

Pretende-se desenvolver o *software* básico disponível de forma a tornar o sistema completamente operacional para medidas de campo, nomeadamente:

- Implementar um AGC (coerente ou não) de forma a manter os parâmetros da malha constantes mesmo que a amplitude do sinal varie, ou seja, que a SNR se degrade;
- Implementar algoritmos para aquisição rápida do sinal em "*Power On*";
- Estudar, avaliar e implementar técnicas alternativas de seguimento do sinal que possam contribuir para a manutenção de sincronismo ou estender a gama dinâmica;
- Definir políticas de administração da malha para SNR degradadas;
- Filtrar e transferir em tempo real as componentes cartesianas da amplitude do sinal para o PC anfitrião;
- Avaliar o desempenho do sistemaem situações tão realistas quanto possível;
- Explorar as facilidades de interface do *Matlab* com o *Code Composer Studio* para melhorar a produtividade do *software* subsequente (opcional).

Conseguir estes objectivos melhoraria significativamente as potencialidades experimentais não só pelas vantagens descritas anteriormente bem como pelo facto de que minoraria significativamente os esforços de desenvolvimento do *hardware* analógico. O *hardware analógico* é bastante complexo nestes receptores principalmente devido às necessidades dos circuitos de sincronização, detecção síncrona e esquemas de síntese coerente de osciladores locais.

2. Malhas de Sincronização

A medição de um sinal CW de baixa CNR com máxima sensibilidade que varie lentamente em frequência, exige que este seja sincronizado com uma frequência de referência do receptor. Poderão então ser aplicados filtros passa-banda estreitos e proceder-se à detecção síncrona. Como se pode ver na Figura 2.1, o funcionamento de um detector síncrono consiste basicamente na multiplicação do sinal de entrada por um oscilador local sincronizado com o sinal de entrada. A saída do detector corresponde às componentes cartesianas, I_n e Q_n, obtidas pela decomposição de um fasor nas suas componentes ortogonais relativas ao oscilador local. Esta aproximação permite calcular a amplitude do sinal (quadrando, somando e extraindo a raiz quadrada) bem como determinar a sua fase em relação a outro coerente com este. Tal poderá ser no nosso caso, a fase entre o sinal copolar recebido do satélite e o sinal crosspolar.

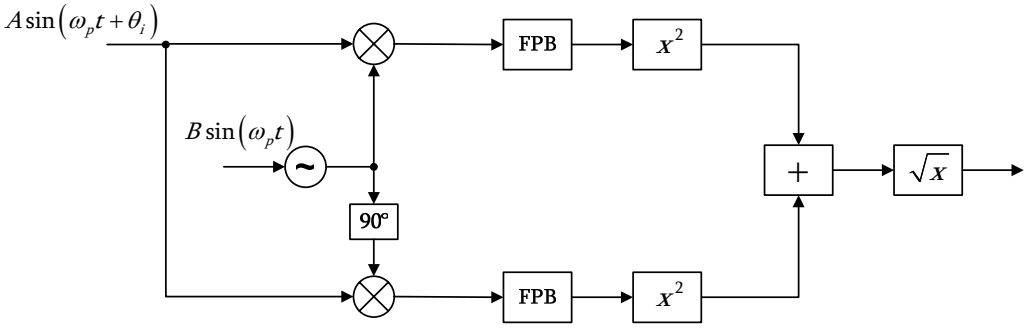

Figura 2.1 – Diagrama de blocos de um detector coerente.

As componentes cartesianas DC do sinal podem ser filtradas num filtro passa-baixo de largura de banda reduzida (filtro de pré-detecção) e de seguida a potência e a fase do sinal determinadas. Após a estimativa da potência o sinal pode ainda ser filtrado (filtro pós-detecção) contudo o ruído detectado já não poderá ser removido conduzindo a um *offset* na amplitude.

Os erros de medida (sistemáticos e aleatórios) usando uma PLL encontram-se descritos em [1, cap. 2, pág. 2] e são função da CNR, da largura de banda da PLL e da largura de banda dos filtros pré e pós-detecção. Naturalmente a estimativa da potência do sinal poderá ser feita no domínio da frequência somando, por exemplo, as potências das riscas espectrais mais importantes do sinal. Contudo, também neste caso a potência de ruído na largura de banda considerada irá causar um *offset*.

Abordam-se neste capítulo as malhas de captura de fase (PLL – *Phase Locked Loop*) e de frequência (FLL – *Frequency Locked Loop*) bem como a necessária malha de controlo automático de ganho (AGC). A avaliação do desempenho com estimação espectral é igualmente possível com o *software* desenvolvido. Porém este tópico não foi abordado, apesar de ser uma aproximação mais directa embora oferecendo uma menor

sensibilidade e a técnica necessitar de algumas modificações para medir o denomindado sinal crosspolar.

2.1. PLL – *Phase Locked Loop*

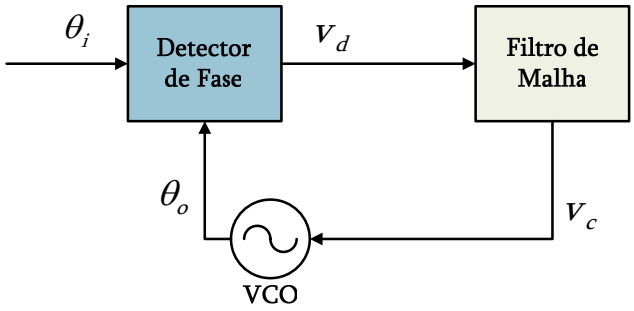

Figura 2.2 – Diagrama de blocos de uma PLL ou malha de seguimento de fase.

Uma PLL ou malha de seguimento de fase, é um sistema de realimentação constituído por três elementos essenciais que se podem observar na Figura 2.2, nomeadamente um detector de fase (PD – *Phase Detector*), um filtro de malha e um oscilador local com frequência controlada por tensão (VCO – *Voltage Controlled Oscillator*). O sinal à saída do filtro de malha é usado para sincronizar a frequência ou fase instantânea do VCO com a frequência ou fase instantânea do sinal de entrada. Uma vez em sincronismo qualquer desvio de frequência do sinal de entrada é compensado com o desenvolvimento de uma tensão de correcção no detector de fase.

O objectivo da PLL é efectuar o sincronismo do sinal, ou por outras palavras, manter iguais em termos médios as frequências de um sinal (poderá ser a portadora num sistema de telecomunicações) e de um oscilador local. No entanto, este sinal vem geralmente corrompido por ruído aditivo. Desta forma, a PLL terá que ser dimensionada de forma conveniente para rejeitar o mais possível este ruído (o ruído à saída do filtro de malha modula o VCO). Esta capacidade de funcionar correctamente com sinais corrompidos por ruído constitui a grande vantagem das PLL.

Uma outra contrariedade do sinal é ter a sua potência espalhada numa determinada largura de banda (ruído de fase), ou seja, assemelha-se a um sinal sinusoidal com uma fase aleatória $\phi(t)$:

$$A(t) = A_o \sin\left(w_c t + \theta_i + \phi_n(t)\right). \tag{2.1}$$

O detector de fase mede a diferença de fase entre o sinal periódico de entrada e o sinal à saída do VCO, apresentando à saída uma tensão correspondente a esta diferença de fase. Esta tensão é posteriormente filtrada pelo filtro de malha e aplicada ao VCO. Este por sua vez vai alterar a frequência de oscilação de forma a obter a menor diferença de fase entre o sinal de entrada e o do oscilador local. Para eliminar o ruído, o sistema serve-se do filtro de malha que funciona como um filtro passa-baixo rejeitando

produtos de mistura do detector de fase (frequência dupla fundamentalmente) e eventualmente ruído que acompanhe o sinal a sincronizar.

Todavia, isto não é tão simples como se poderia pensar. Se o sinal a sincronizar for estável na frequência, para eliminar o ruído que pode ser elevado, basta efectuar a média num longo período de tempo (usar filtro de malha com uma longa constante de tempo), uma vez que o oscilador local não necessita de muita informação para eliminar o ruído e obter a frequência desejada. Porém, quando o sinal tem a sua potência espalhada no espectro, terão que se ter em conta alguns compromissos no que se refere denominada à largura de banda de ruído da PLL (B_L). Esta tem que ser suficientemente larga, para que o VCO possa acompanhar com agilidade as flutuações mais rápidas de frequência/fase do sinal a sincronizar. No entanto, a largura de banda também não pode ser muito maior que a necessária, dado o consequente aumento de ruído térmico que degradaria o desempenho do sistema. Por outro lado, se esta for muito reduzida, está a desperdiçar-se potência do sinal. Assim, na implementação de uma PLL, é necessário um adequado dimensionamento da referida largura de banda de ruído a qual depende das características do detector de fase, do VCO e do filtro de malha.

2.1.1. Detector de Fase

Assumindo que o detector de fase é linear, este produz à saída uma tensão v_d, com características lineares e que se repete em cada 2π radianos, dada por:

$$v_d = K_d\left(\theta_i - \theta_o\right) = K_d\theta_d\,, \tag{2.2}$$

onde θ_d representa a diferença entre a fase de entrada, θ_i, e a fase de saída do VCO, θ_o, e onde K_d é o factor de ganho do detector de fase em V/rad.

Reescrevendo a equação (2.2) em termos de transformadas de Laplace, obtém-se:

$$V_d(s) = K_d\left[\theta_i(s) - \theta_o(s)\right] = K_d \cdot \theta_d(s), \tag{2.3}$$

onde $\theta_d(s)$ representa a diferença entre a transformada de Laplace da fase de entrada, $\theta_i(s)$, e a transformada da fase de saída do VCO, $\theta_o(s)$.

2.1.2. Filtro de Malha

O filtro de malha é bastante influente sobre o comportamento dinâmico da PLL. O filtro é de natureza passa-baixo, para atenuar o ruído e eliminar componentes de alta-frequência à saída do detector de fase tal como a frequência soma. Este filtro funciona portanto como integrador, efectuando a média do sinal v_d ao longo do tempo e gerando um sinal de controlo v_c que é depois utilizado para estabelecer a frequência do VCO, mantendo o sistema em sincronismo. Em termos de transformadas de Laplace, o funcionamento do filtro pode ser descrito por:

$$V_c(s) = F(s) \cdot V_d(s), \tag{2.4}$$

onde $F(s)$ representa a função de transferência do filtro e onde $V_c(s)$ e $V_d(s)$ representam respectivamente as transformadas de Laplace dos sinais v_c e v_d.

2.1.3. VCO

Existem dois tipos de VCO's, nomeadamente os lineares, que são utilizados em PLL's lineares e em PLL's que funcionam para frequências elevadas, na ordem das centenas de MHz e GHz, e os digitais, que são utilizados em PLL's digitais. Nos VCO's lineares, os elementos controlados por tensão são geralmente díodos varicap integrados num circuito LC, em que se faz variar a frequência instantânea do oscilador variando a capacidade do díodo através de uma tensão de controlo. Os VCO's digitais são geralmente circuitos astáveis e utilizados até algumas dezenas de MHz.

Geralmente, os requisitos desejados para o VCO entram em conflito mútuo, sendo por isso necessário encontrar compromissos entre estes. De entre estes requisitos destacam-se a estabilidade, a linearidade da frequência em função da tensão de controlo, um elevado factor de ganho K_o e extensas gamas de tensões à entrada e de variação de frequência. A estabilidade está em oposição com todos os restantes requisitos, pelo que para se conseguir que um destes se verifique é necessário algum sacrifício na estabilidade do VCO.

A característica típica de um VCO é uma curva crescente em que o declive é o factor de ganho ou a constante do VCO, K_o, cujas dimensões são em Hz/V. Aqui a frequência do oscilador ω_o é uma função da tensão de controlo v_c em torno de uma frequência central. Esta função não necessita de ser linear, mas tal simplificaria a implementação e análise do comportamento da PLL.

O desvio do VCO em torno da frequência central é dado por:

$$\Delta\omega_o = K_o v_c . \tag{2.5}$$

Sabendo que a frequência é a derivada em ordem ao tempo da fase, a equação anterior pode ser reescrita como:

$$\frac{d\theta_o}{dt} = K_o v_c . \tag{2.6}$$

Efectuando as correspondentes transformações de Laplace tem-se que:

$$s \cdot \theta_o(s) = K_o V_c(s) \Leftrightarrow \theta_o(s) = \frac{K_o V_c(s)}{s}, \tag{2.7}$$

pelo que se pode concluir que a fase à saída do VCO, θ_o, é proporcional ao integral da tensão de controlo v_c.

2.1.4. Função de Transferência

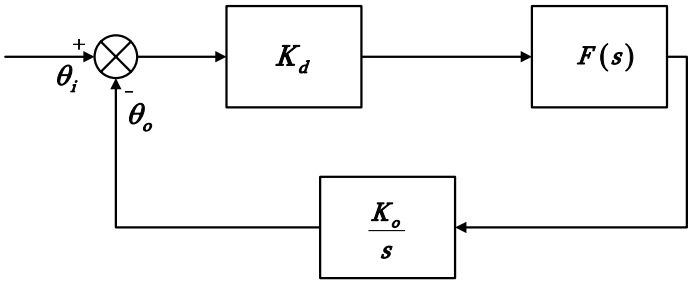

Figura 2.3 – Diagrama de blocos da PLL e correspondentes funções de transferência.

Na Figura 2.3 está representado um diagrama da malha da PLL, cujas funções de transferência de cada bloco estão de acordo com as respectivas equações (2.3), (2.4) e (2.7). Como se pode facilmente observar, a função de transferência em malha aberta é dada por:

$$G(s) = K_d K_o F(s) \tag{2.8}$$

e logo, a função de transferência em malha fechada é dada por:

$$H(s) = \frac{G(s)}{1+G(s)} = \frac{K_d K_o F(s)}{s + K_d K_o F(s)}. \tag{2.9}$$

O factor $K_o K_d$ é conhecido como o ganho da malha, sendo as suas dimensões em frequência.

Da equação (2.8), pode-se concluir que o ganho em DC em malha aberta é dado por:

$$K_v = s \cdot G(s)\big|_{s=0} = K_d K_o F(0). \tag{2.10}$$

2.1.4.1. Filtro Activo de 2ª Ordem

Um dos filtros de malha analógicos mais utilizados é um filtro activo e está representado na Figura 2.4. Este filtro activo de 2ª ordem apresenta um melhor desempenho ao nível do seguimento, quando comparado com um filtro passivo.

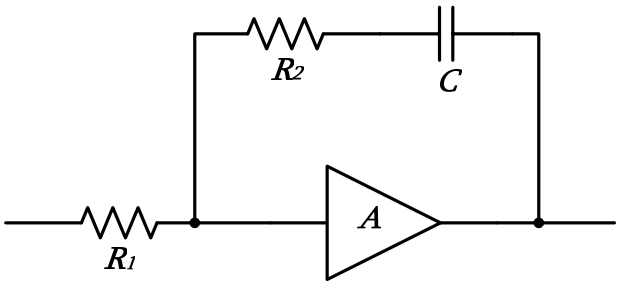

Figura 2.4 – Filtro activo de 2ª ordem.

A função de transferência deste filtro é dada por:

$$F(s) = \frac{A(sCR_2 + 1)}{sCR_2 + 1 + (1 - A)(sCR_1)}. \tag{2.11}$$

Para um ganho A do amplificador elevado, obtém-se a seguinte aproximação para a função de transferência:

$$F(s) \approx \frac{sCR_2 + 1}{sCR_1} = \frac{s\tau_2 + 1}{s\tau_1}, \tag{2.12}$$

onde τ_1 e τ_2 representam as constantes de tempo, podendo-se concluir que a função de transferência do filtro apresenta um factor proporcional (R_2/R_1) e um factor integrador ($1/sCR_1$). Repare-se que este filtro possui um ganho infinito para DC, ou seja, a sua tensão de saída pode ser qualquer para uma tensão à entrada (saída do detector de fase) de valor médio nulo. Na prática, em muitos detectores de fase isto significa que o VCO estará em quadratura com o sinal de entrada.

Supondo então que o ganho do amplificador é elevado, a função de transferência em malha fechada, é dada por:

$$H(s) = \frac{K_d K_o (s\tau_2 + 1)/\tau_1}{s^2 + s(K_d K_o \tau_2/\tau_1) + K_d K_o/\tau_1}, \tag{2.13}$$

podendo no entanto ser reescrita como:

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{2.14}$$

onde ω_n é a frequência natural e ζ é o coeficiente de amortecimento, dados por:

$$\omega_n = \sqrt{\frac{K_d K_o}{\tau_1}}, \tag{2.15}$$

$$\zeta = \frac{\tau_2}{2}\sqrt{\frac{K_d K_o}{\tau_1}}. \tag{2.16}$$

2.1.5. Largura de Banda do Ruído

Considere-se que o sinal à entrada da PLL é um sinal sinusoidal que vem corrompido por ruído, dado por:

$$A\cos\left[\omega_c t + \phi_s(t) + \phi_n(t)\right] + n(t), \tag{2.17}$$

em que $\phi_s(t)$ representa o sinal de fase, $\phi_n(t)$ representa o ruído de fase e $n(t)$ representa o ruído de amplitude.

O ruído de amplitude introduz flutuações na fase do sinal, ou seja, ruído de fase já que o ruído causa de facto incerteza na passagem por zero do sinal sinusoidal.

A potência (ou variância) do ruído de fase equivalente é dada por:

$$\overline{\theta_{ni}^2} = \frac{P_n}{2P_S}. \tag{2.18}$$

A razão sinal ruído à entrada da PLL, SNR_i, é definida como a razão entre P_s e P_n, ou seja:

$$SNR_i = \frac{P_n}{P_S} \Rightarrow \overline{\theta_{ni}^2} = \frac{1}{2SNR_i}. \tag{2.19}$$

Por outro lado a densidade espectral de potência de um determinado tipo de ruído depois de filtrado por um filtro cuja função de transferência é $H(f)$, é dada por:

$$S_o(f) = \left|H(f)\right|^2 \cdot S_i(f), \tag{2.20}$$

em que $S_I(f)$ e $S_o(f)$ representam a densidade espectral de potência do ruído à entrada e à saída do filtro, respectivamente.

Atendendo que a PLL é efectivamente um filtro caracterizado por uma função de transferência, $H(f)$, então é possível caracterizar o ruído de fase à saída da PLL (saída do VCO) através da seguinte expressão:

$$\theta_{no}^2(f) = \left|H(f)\right|^2 \cdot \theta_{ni}^2(f), \tag{2.21}$$

em que $\theta_{no}(f)$ representa a densidade espectral de potência do ruído de fase do sinal de saída da PLL. Finalmente é possível determinar a potência do ruído de fase do sinal de saída. Esta é dada por:

$$\theta_{no}(f) = \int_{-\frac{B}{2}}^{\frac{B}{2}} \left|H(f)\right|^2 \cdot \theta_{ni}(f)\,df = \phi \cdot B_L \Rightarrow B_L = \int_{-\frac{B}{2}}^{\frac{B}{2}} \left|H(f)\right|^2 df, \tag{2.22}$$

em que B_L é designado por largura de banda de ruído.

Segundo [2, pág. 20], para um filtro $H(f)$ de 2ª ordem tem-se que:

$$B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right). \tag{2.23}$$

A largura de banda de ruído é proporcional à frequência natural, ω_n, dependendo também do amortecimento ζ. A Figura 2.5 ilustra a variação da largura de banda, normalizada em relação à frequência natural, em função do amortecimento. Verifica-se que, para um determinado ω_n, B_L apresenta um mínimo para $\zeta = 0.5$.

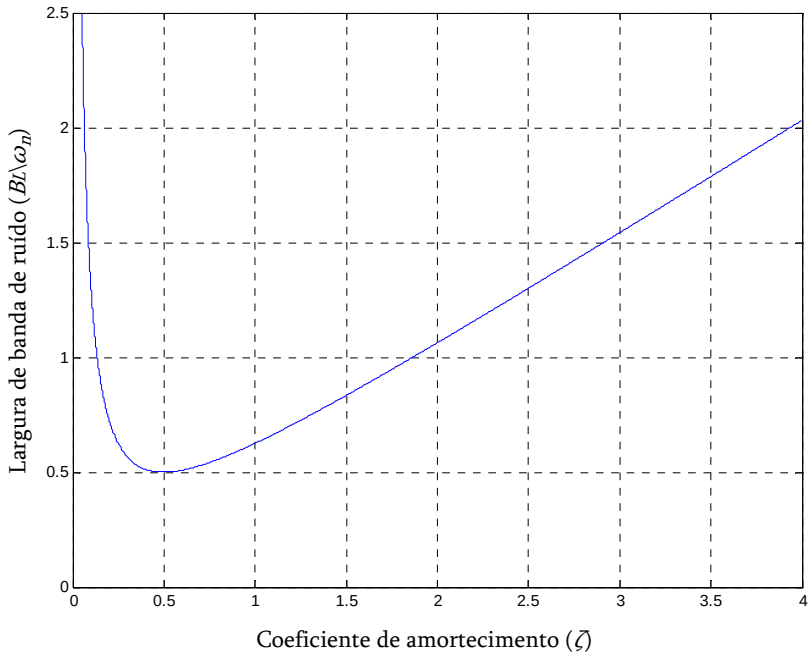

Figura 2.5 – Gráfico da largura de banda de ruído para o filtro de 2ª ordem.

Considere-se o exemplo de uma malha, dimensionada com valores típicos de *hardware* para uma implementação analógica, cujo diagrama de Bode está representado na Figura 2.6.

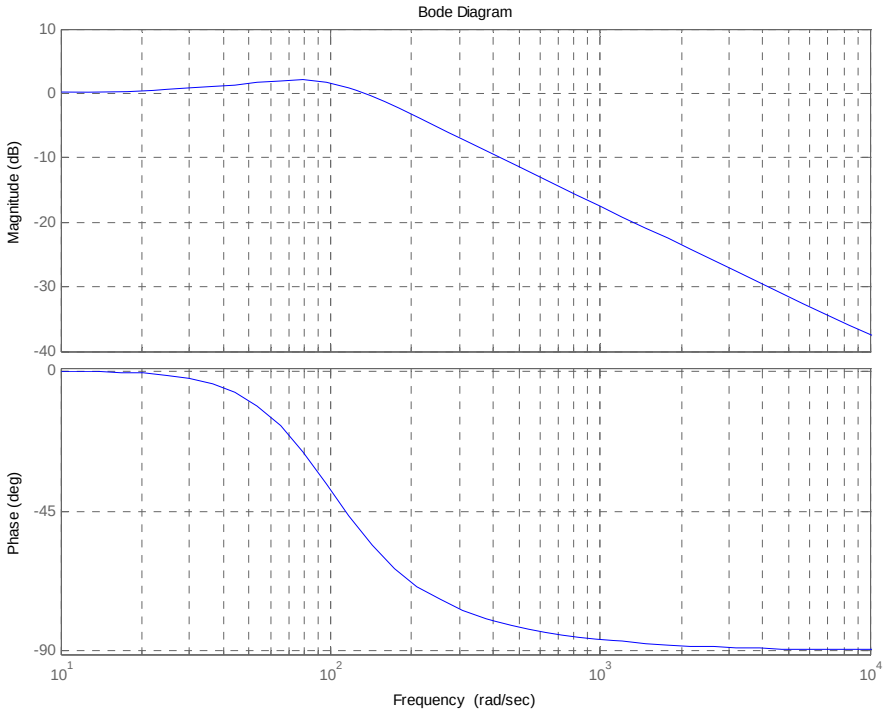

Figura 2.6 – Diagrama de bode da PLL, com K_d = 0.5V/rad, K_o = 5KHz/V, B_L = 50Hz e ζ = 0.707.

2.1.6. Limiar de Ruído

À medida que a relação sinal-ruído do sinal à entrada se reduz, o *jitter* de fase à saída aumenta. Se o erro de fase exceder os limites de funcionamento do detector de fase de forma sistemática é de esperar que a malha perca o sincronismo.

O erro de fase devido ao ruído é uma quantidade estatística geralmente descrita pelo seu valor médio quadrático (*rms*). No entanto, por mais insignificante que seja o ruído, há sempre a possibilidade dos picos de ruído excederem largamente o seu *rms* e consequentemente dos limites do detector de fase serem também ultrapassados. Como é óbvio, a probabilidade disto acontecer é tanto maior quanto maior for o ruído. Por outras palavras, para sinais fortes a probabilidade da malha perder o sincronismo devido ao ruído é negligível, mas à medida que o ruído aumenta esta probabilidade também aumenta até ao ponto em que é impossível manter o sincronismo.

Surge então o conceito de "*loop threshold*", que consiste num valor de limite de SNR_L para o qual a malha apresenta um desempenho desejado. Este é portanto um conceito flexível, podendo o desempenho da PLL variar de acordo com os critérios definidos. O critério mais evidente pode ser descrito como o valor mínimo de SNR_L, abaixo do qual a malha perde o sincronismo. Porém muitos outros critérios podem ser considerados, como por exemplo, o valor limite de SNR_L que permite manter o *jitter* de fase à saída inferior a um determinado nível.

Para poder definir estes critérios, é necessário conhecer a variação do *jitter* de fase à saída com a relação sinal-ruído na malha. Assim, tém-se a seguinte aproximação linear:

$$\overline{\theta_{no}^2} = \frac{1}{2SNR_L}, \tag{2.24}$$

que só válida para valores de SNR_L > 10dB. O valor de 10dB é normalmente utilizado na aplicação em causa neste trabalho, já que valores superiores conduzirão a erros significativos na detecção coerente da amplitude, que depende desta flutuação de fase e de outros parâmetros.

2.1.7. Seguimento Linear

Para saber mais sobre o seguimento numa PLL em sincronismo, não faz sentido analisar o erro de frequência, já que este não é suposto existir, mas sim o erro de fase θ_e que resulta de uma determinada fase à entrada θ_i. Um bom desempenho de seguimento depende sempre de um erro de fase reduzido.

Os detectores de fase mais comuns apresentam uma característica sinusoidal, gerando à saída um sinal de controlo que se relaciona com o erro de fase por:

$$e_d = K_d \sin\theta_e . \tag{2.25}$$

Considerando que o erro de fase é suficientemente pequeno, ou seja, $\theta_e \approx \sin\theta_e$, então a aproximação linear (2.24) é válida e a malha pode ser vista como operando linearmente.

Como é óbvio, inicialmente não é de esperar que a frequência do sinal de entrada coincida exactamente com a frequência inicial do VCO (*zero control voltage* – frequência central). O erro de fase daí resultante é designado por erro de fase estático ou erro de velocidade e, no caso de uma malha linear, é dado por:

$$\theta_v = \frac{\Delta\omega}{K_v}, \tag{2.26}$$

onde $\Delta\omega$ é a diferença de frequência entre o sinal de entrada e o VCO.

Analisando a equação anterior é possível observar que quanto maior for o ganho DC da malha, menor será o erro de fase estático e esta é uma característica bastante importante no seguimento. Contudo, ao contrário do que se possa pensar, não é possível tornar o erro de fase estático indeterminadamente pequeno, bastando para tal aumentar o ganho K_v. Como se verá mais adiante, é absolutamente necessária a existência de um erro mínimo, para que a PLL consiga adquirir o sincronismo.

2.1.8. Manutenção de Sincronismo (*Hold-in*)

A aproximação linear considerada na secção anterior, vai perdendo validade à medida que o erro de fase aumenta, até um ponto em que a malha acaba por perder o sincronismo. Nesta situação, torna-se então necessário averiguar quais as condições limite para as quais mantém o sincronismo.

A expressão correcta para o erro de fase estático, tendo em conta um detector de fase de característica sinusoidal, é então dada por:

$$\sin\theta_v = \frac{\Delta\omega}{K_v}. \tag{2.27}$$

Como o seno varia entre -1 e 1, a gama de *hold-in* é definida por:

$$\Delta\omega = \pm K_v. \tag{2.28}$$

Caso $|\Delta\omega| > K_v$, não existiria solução para a equação (2.27) e consequentemente a PLL perderia o sincronismo, sendo que o sinal à saída do detector de fase, em vez de DC, se transformaria num *beat-note* (ver secção 2.1.9). Basicamente significa que a máxima diferença de fase não produz uma tensão suficiente para manter a frequência do VCO igual à do sinal de entrada.

No entanto, a gama de *hold-in* não pode tomar valores arbitrários variando o ganho da malha, como se afirmou na secção anterior. Se este ganho for demasiadamente elevado, os limites de funcionamento dos restantes elementos da malha seriam ultrapassados e esta perderia o sincronismo. Por outras palavras, é necessário haver um

erro de fase não nulo para que seja gerado um sinal de controlo capaz de "empurrar" o VCO em direcção à frequência do sinal de entrada.

Relativamente ao erro de aceleração numa malha de 2ª ordem, também designado por erro dinâmico de seguimento e que é devido a uma alteração não instantânea de frequência à entrada, este é dado por:

$$\sin\theta_a = \frac{\Delta\omega}{\omega_n}, \tag{2.29}$$

onde ω_n é a frequência natural da malha de 2ª ordem.

Por analogia com a definição da equação (2.28), a máxima taxa de variação de frequência à entrada para a qual a malha se mantém em sincronismo, é dada por:

$$\Delta\omega = \omega_n^2. \tag{2.30}$$

Como se verá mais adiante, a aquisição de sincronismo com taxas de variação de frequência próximas de ω_n^2 é muito difícil ou mesmo impossível. A situação de variação de frequência do sinal não é esperada nestas aplicações pois o satélite não se move significativamente.

Surge no entanto uma questão importante, que consiste na possibilidade do erro de fase transiente em resposta a uma variação de frequência ser superior ao erro de fase estático. Desta forma, mesmo que a malha esteja na gama de *hold-in*, é possível que o erro de fase transiente possa provocar a perda de sincronismo.

A cada ciclo do sinal de entrada, corresponde um único ciclo à saída, pelo que se o oscilador, mesmo que por breves instantes falhar algum ciclo, a malha acaba por perder o sincronismo. No entanto, segundo Rue and Lux [2, pág. 37], uma malha de 2ª ordem com ganho infinito nunca perderia permanentemente o sincronismo. Eventualmente, o que poderia acontecer seria a perda de alguns ciclos mas posteriormente o sincronismo seria readquirido. Mas para tal acontecer, seria necessário aplicar à entrada um degrau que superasse um determinado limite designado por frequência de *pull-out*.

De acordo com simulações realizadas por Viterbi [2, pág. 37], a frequência de *pull-out* para uma malha de 2ª ordem de ganho elevado pode ser aproximada linearmente por:

$$\Delta\omega_{PO} = 1.8\omega_n(\zeta + 1), \tag{2.31}$$

para a gama de ζ estudada por Viterbi.

Caso $|\Delta\omega| < \Delta\omega_{PO}$, o erro de fase transiente é suficientemente pequeno para garantir que a malha se mantém em sincronismo. Mas por outro lado, se $|\Delta\omega| > \Delta\omega_{PO}$, a malha perde ciclos durante um determinado período de tempo, mas posteriormente volta a adquirir o sincronismo.

Ao contrário do que se poderia pensar, quando $|\Delta\omega|=\Delta\omega_{PO}$, o pico do erro de fase não corresponde a 90° mas sim a 180°. De acordo com os estudos de Viterbi, o erro de fase cresce muito rapidamente ao alcançar os 90°.

Existe ainda um limite superior designado por frequência de *dropout*, acima do qual a malha perderia o sincronismo mas não o voltaria a readquirir.

Todavia, é de realçar o facto de que, para malhas de 2ª ordem com ganho moderado, o desempenho se degradaria comparativamente com a situação ideal de ganho infinito, sendo que os valores para os limites de frequência descritos anteriormente se reduziriam.

De salientar ainda que até agora se consideraram malhas livres de ruído. Na presença de ruído será de esperar que efeitos semelhantes aos descritos anteriormente se repitam. Como é óbvio, os desvios de frequência no NCO são tanto maiores quanto menor for a relação sinal-ruído na malha. Devem-se então evitar as situações mais preocupantes que surgem quando o sinal de entrada se atenua bastante.

Assim, quando a SNR atingir valores demasiado baixos, uma possível solução para esta questão baseia-se no "congelamento" do NCO ou, por outras palavras, na abertura da malha mantendo a frequência do NCO num determinado valor fixo. Uma vez que não se esperam variações de frequência rápidas no sinal de entrada, este valor poderia corresponder a uma média das frequências configuradas no NCO, durante o tempo em que a malha está em sincronismo com uma boa SNR. Desta forma, quando o sinal recuperasse fechar-se-ia de novo a malha e seria de esperar a rápida reaquisição de sincronismo.

2.1.9. Aquisição de Sincronismo

Inicialmente não se poderá esperar que a malha esteja em sincronismo, pelo que é importante conhecer as diversas situações de aquisição.

O aspecto fundamental desta questão reside na diferença inicial entre a frequência do sinal de entrada e a frequência do oscilador, sendo que a malha apresenta então vários comportamentos correspondentes a diferentes gamas de frequência.

Se inicialmente esta diferença for próxima da largura de banda da PLL, o sincronismo é obtido de forma quase instantânea. A diferença máxima que permite esta aquisição rápida é a frequência de *lock-in* – $\Delta\omega_L$.

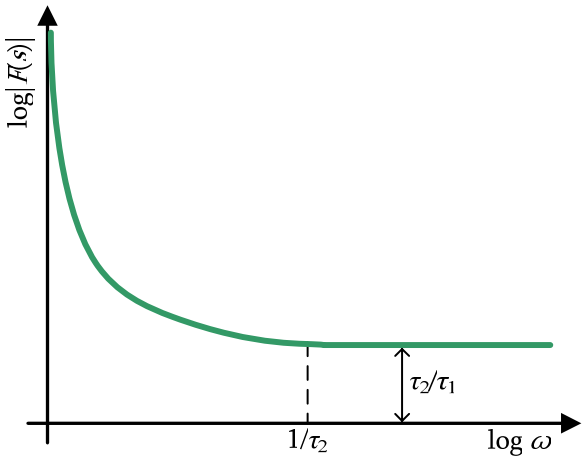

Figura 2.7 – Resposta em frequência de um filtro activo de 2ª ordem.

De acordo com [2, págs. 41-43], a frequência de *lock-in* numa malha com um filtro activo de 2ª ordem é igual ao seu ganho total às altas frequências. Na Figura 2.7, onde está representada a resposta em frequência deste tipo de filtros, é possível observar que o ganho do filtro às altas frequências corresponde a τ_1/τ_2. Assim, considerando ainda os ganhos dos restantes elementos da malha (detector de fase e VCO), a frequência de *lock-in* é dada por:

$$\Delta\omega_L = \frac{K_o K_d \tau_2}{\tau_1}. \tag{2.32}$$

Utilizando as equações (2.15) e (2.16), a frequência de *lock-in* pode ser expressa em função dos parâmetros da malha:

$$\Delta\omega_L = 2\zeta\omega_n. \tag{2.33}$$

Se a diferença de frequência entre o sinal de entrada e o VCO estiver contida na largura de banda da malha, é de esperar que este adquira imediatamente o sincronismo sem falhar nenhum ciclo. O tempo necessário para que este *lock-in* ocorra é dado por:

$$T_L = \frac{1}{\omega_n}. \tag{2.34}$$

De referir ainda que numa malha de 2ª ordem é geralmente mais fácil manter o sincronismo do que adquiri-lo rapidamente. Isto acontece porque o ganho K_V é superior à frequência natural ω_n, pelo que a frequência de *hold-in* é também superior à frequência de *lock-in*.

Todavia, neste tipo de malhas é também possível a aquisição de sincronismo sem qualquer tipo de ajuda, mesmo para diferenças de frequência superiores a $\Delta\omega_L$ e portanto muito superiores à sua largura de banda. Existe um limite de frequência para o qual este tipo de aquisição é possível, que é designado por frequência de *pull-in*. De acordo com Richman [2, págs. 45-46], em malhas de 2ª ordem com ganho elevado, este limite pode ser aproximado por:

$$\Delta\omega_{PI} = 2\sqrt{\zeta\omega_n K_v}\ . \qquad (2.35)$$

Dentro desta gama de *pull-in*, a frequência do oscilador vai-se aproximando lentamente da frequência de entrada, até que a malha adquire o sincronismo desde que eventualmente não seja interrompido até lá. Este comportamento é explicado pelo facto de que antes da malha entrar em sincronismo, a saída do detector de fase é um sinal do tipo *beat-note* à diferença de frequência entre a entrada e o oscilador. Apesar deste *beat-note* ser reduzido em amplitude (por um factor τ_2/τ_1) no filtro de malha, este é suficiente para modular o oscilador em frequência. Neste sentido, a saída do detector de fase é o produto de uma onda sinusoidal por uma onda modulada em frequência.

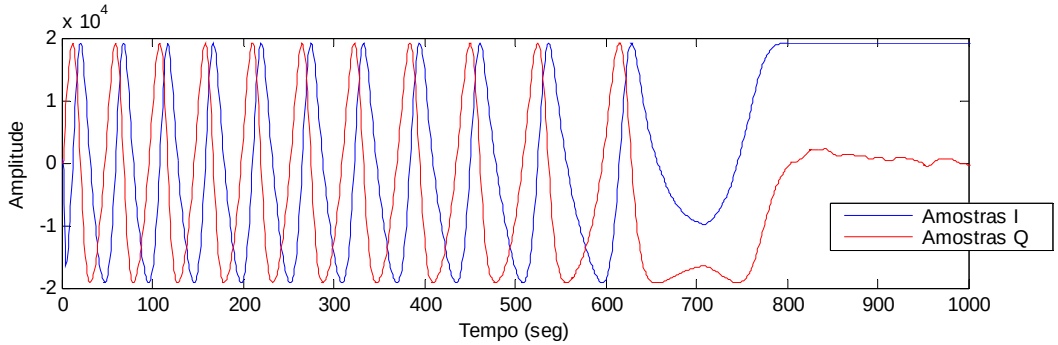

Figura 2.8 – Forma de onda característica de um sinal do tipo *beat-note*.

Na Figura 2.8 está representada a forma de onda do *beat-note*. Este sinal foi observado num dos testes realizados no projecto anterior, quando a diferença de frequência entre o NCO e o sinal de entrada era significativa, mas só se observava até ao instante em que a malha adquiria o sincronismo.

O *beat-note*, além de apresentar uma característica não sinusoidal, apresenta excursões positivas e negativas não iguais em área. Consequentemente a sua componente DC não é nula, o que se revela de fundamental importância já que o filtro de malha de 2ª ordem contém um integrador que responde a essa componente DC com um sinal progressivamente crescente. À medida que a tensão DC se desenvolve, vai promovendo o desvio de frequência do VCO em direcção à frequência de entrada até que a malha entra em sincronismo. A presença desta componente DC é portanto essencial para a ocorrência do *pull-in*. Na prática o que poderá acontecer é algum *offset* dos amplificadores operacionais no filtro da malha contrariar o valor DC gerado neste batimento, levando à saturação definitiva do integrador, enorme desvio do VCO e logo à impossibilidade do sincronismo ser alcançado espontaneamente.

De acordo com Viterbi [2, pág. 46], para uma determinada diferença inicial de frequência $\Delta\omega$ dentro da gama de *pull-in*, o tempo necessário para a PLL adquirir o sincronismo pode ser aproximado por:

$$T_{PI} \approx \frac{(\Delta\omega)^2}{2\zeta\omega_n^3} \quad [\mathrm{s}], \qquad (2.36)$$

sendo esta aproximação apenas válida para valores intermediários da gama de *pull-in*, ou seja, nem muito próximos de $\Delta\omega_L$ nem de $\Delta\omega_{PI}$. Este corresponde ao tempo que a PLL demora a reduzir a diferença de frequência, desde $\Delta\omega$ até $\Delta\omega_L$, já que a partir do instante em que este último valor é atingido, é de esperar que a malha adquira o sincronismo quase instantaneamente.

Substituindo a equação (2.23) da largura de banda de ruído na equação anterior, obtém-se a expressão particular para o tempo de *pull-in* numa malha de ganho elevado, mas em função da largura de banda:

$$T_{PI} \approx \frac{(\Delta\omega)^2\left(\zeta+\frac{1}{4\zeta}\right)^3}{16\zeta B_L^3} = \frac{\pi^2(\Delta f)^2\left(\zeta+\frac{1}{4\zeta}\right)^3}{4\zeta B_L^3} \quad [\mathrm{s}]. \tag{2.37}$$

Analisando a equação anterior pode-se afirmar que quanto menor for a largura de banda de ruído da PLL, maior será o tempo de *pull-in*. Para se ter uma ideia deste efeito, repare-se no seguinte exemplo. Para uma diferença inicial de frequência de Δf = 500Hz e com ζ = 0.707, se B_L = 50Hz, o tempo para a PLL adquirir o sincronismo seria de aproximadamente 8.3s. Por outro lado, mantendo os mesmos valores de Δf e ζ mas com B_L = 10 Hz, então o tempo de *pull-in* rondaria os 17m20s. Isto permite concluir que para malhas de banda estreita, o tempo de *pull-in* aumenta drasticamente, o que pode ser intolerável para muitas aplicações.

Recapitulando, as gamas de frequência para a aquisição de sincronismo são as seguintes:

- Gama de *lock-in* ($|\Delta\omega|<\Delta\omega_L$) – aquisição rápida de sincronismo ($1/\omega_n$);
- Gama de *pull-in* ($\Delta\omega_L<|\Delta\omega|<\Delta\omega_{PI}$) – aquisição de sincronismo é muito provável mas mais lenta;
- Caso $|\Delta\omega|<\Delta\omega_{PI}$ – aquisição de sincronismo é impossível ou então requer demasiado tempo (a utilização de métodos de auxílio à aquisição é imperativa).

Todavia deve-se salientar o facto de que, tal como acontece com a manutenção do sincronismo (*hold-in*), a aquisição do sincronismo é também afectada fortemente pelo ruído. Consequentemente, os valores referidos anteriormente para as gamas de aquisição de sincronismo não devem ser considerados como valores exactos mas sim como valores meramente indicativos.

Será então de esperar que a frequência de *lock-in* diminua bastante devido às perturbações introduzidas pelo ruído, o que vem realçar a necessidade do sistema arrancar com a frequência do NCO bastante próxima da frequência do sinal de entrada. Por outras palavras, o algoritmo de estimação da frequência do sinal de entrada executado no arranque do sistema, tem que ser bastante fiável de forma a produzir uma

estimativa com um erro de frequência bastante inferior ao limite descrito anteriormente para ω_L.

Por outro lado, a frequência a que o NCO é "congelado" (de acordo com o que se referiu na secção 2.1.8) também não se poderá afastar muito da frequência do sinal. Assim, a solução para estimar este valor tem também que ser muito exigente.

2.1.9.1. Técnicas Auxiliares à Aquisição de Sincronismo

Na eventualidade do tempo de *pull-in* ser demasiado longo é necessário recorrer a técnicas que acelerem a aquisição de sincronismo.

Os métodos de varrimento estão entre as técnicas mais comuns de auxílio à aquisição rápida de sincronismo e baseiam-se na aplicação de uma rampa à entrada do VCO. Se este método for aplicado correctamente, é de esperar que a malha adquira o sincronismo no instante em que a frequência de varrimento do NCO se aproxime da frequência do sinal de entrada.

Todavia, tal como se referiu na secção 2.1.8, existe uma taxa máxima de variação de frequência à entrada para a qual a malha se mantém em sincronismo. Ora se esta taxa não permitir aa malha manter o sincronismo, então certamente também não permitirá a sua aquisição. Desta forma, a aplicação da rampa no VCO deve ter em consideração que a taxa de variação de frequência resultante, apresenta também um limite máximo dado pela equação (2.30).

No caso de malhas de 2ª ordem, em vez de se aplicar uma rampa directamente no VCO, pode-se aplicar um degrau à entrada do filtro. Como este filtro contém um integrador, seria gerada à sua saída a rampa pretendida e cujo declive (taxa de variação) seria controlado pela amplitude do degrau. No entanto, este método só é exequível se o sistema for capaz de suportar a pequena porção degrau que atravessa o filtro e que surge directamente à entrada do VCO, dando origem a um pequeno salto de frequência logo no início do varrimento. Caso contrário, seria necessário recorrer a um gerador independente para gerar a rampa de varrimento. O processo de varrimento deve ser parado quando houver indicação positiva de que a PLL está a reagir: parar demasiadamente tarde ou demasiadamente cedo pode comprometer o sucesso.

De referir ainda que existem muitas outras técnicas de aquisição de sincronismo, de entre as quais se destacam os métodos que empregam diferentes larguras de banda para aquisição e seguimento, os que utilizam circuitos AGC e ainda os que recorrem a discriminadores de frequência (FLL).

2.1.9.2. Memória da Malha

Considerando uma malha de 2ª ordem em sincronismo e na eventualidade do sinal se perder (atenuar-se fortemente), este tenderá a não se desviar muito da frequência do

sinal de entrada. Se o sinal de entrada tiver uma variação de frequência lenta e o sinal não demore muito a regressar, será de esperar que a malha readquira o sincronismo rapidamente, por *lock-in* ou *pull-in*.

Isto acontece porque a informação de frequência é guardada em forma de carga armazenada no integrador do filtro. Nas situações em que o sinal se atenua fortemente, a malha abre-se e o integrador é descarregado com uma constante de tempo de AR_1C. Assim, pode-se afirmar que a malha apresenta uma memória de frequência, que é tanto maior quanto maior for o ganho A da malha. No caso de um filtro activo, o ganho é mais elevado do que num filtro passivo, pelo que apresenta uma maior capacidade de memória. De referir que numa implementação numérica, tal como num integrador digital, o valor do integrador manter-se-á constante.

Existem sempre algumas não idealidades do *hardware* da malha: desvios, *offsets* e ruído. Apesar da memória da malha estas perturbações são integradas pelo filtro e vão desviando o VCO da frequência correcta. No entanto, existe um ganho DC óptimo que maximiza o tempo de memória da malha, permitindo um equilíbrio entre o efeito da descarga do integrador e os efeitos indesejados destas perturbações. Como é óbvio, qualquer iniciativa tomada para reduzir estas perturbações permite também aumentar o tempo de memória.

2.1.9.3. Indicação de Sincronismo

Uma questão importante reside em saber se a malha já terá ou não adquirido o sincronismo. O detector de sincronismo utiliza um segundo detector de fase mas com uma referência de frequência em quadratura com a usada no detector de fase. Assim teremos à saída um sinal DC quando a PLL estiver em sincronismo. Apesar de ser fácil detectar a aquisição de sincronismo, especialmente quando o sinal de entrada apresenta uma boa relação sinal-ruído, não é possível obter uma indicação de sincronismo instantânea. Para se obter uma indicação fiável face às inteferências introduzidas pelo ruído, é sempre necessário filtrar esta indicação e obter uma média com um determinado tempo de integração. Desta forma há sempre um atraso implícito entre uma indicação positiva de sincronismo e o instante em que a malha o adquiriu.

Essencial em aplicaçõs em que o sinal a sincronizar varia em amplitude é a existência de um circuito que estabilize a amplitude de forma a poder definir-se de forma mais fiável o patamar para detecção do sincronismo.

2.1.9.4. Exemplos Observados de Aquisição de Sincronismo

Na Figura 2.9 estão representados três exemplos de aquisição de sincronismo com diferentes diferenças de frequência, Δf, entre a frequência inicial do NCO (10.7MHz) e o sinal de entrada.

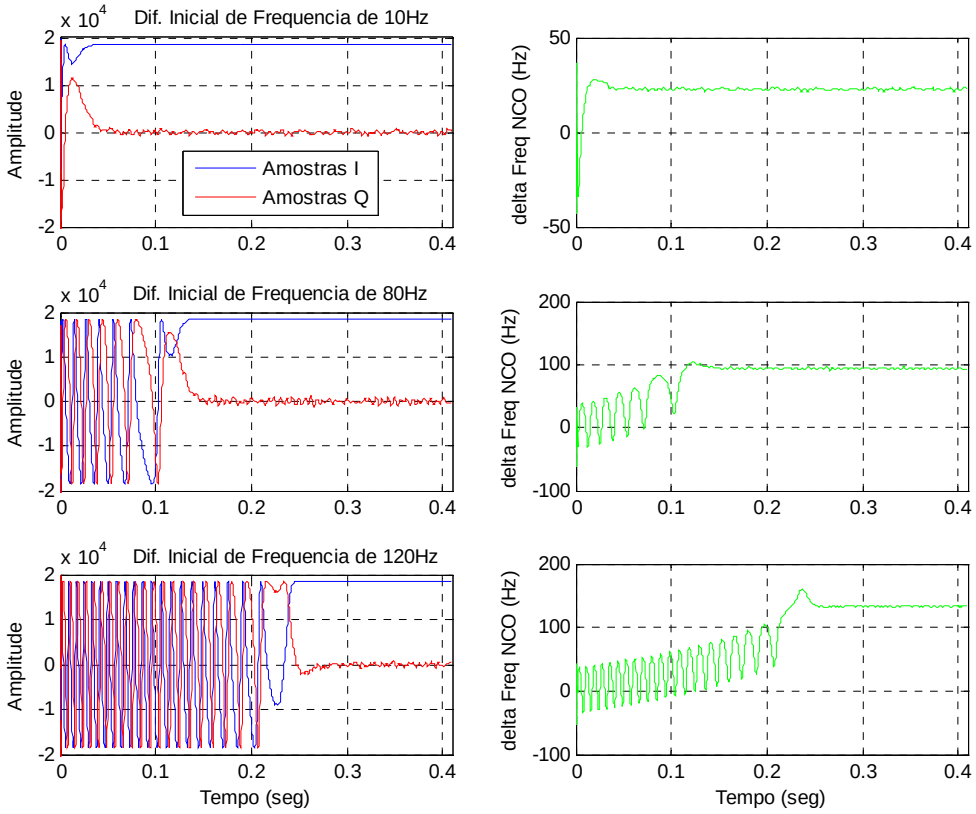

Figura 2.9 – Exemplos de aquisição de sincronismo.

Como se pode observar nos exemplos anteriores, quando a PLL entra em sincronismo, a frequência no NCO estabiliza à volta de uma frequência próxima da do sinal de entrada. Além disso, a componente Q com que é realizada a realimentação da PLL, toma valores próximos de zero, enquanto que a componente I toma um valor DC que corresponde à amplitude do sinal de entrada.

Comparando os exemplos, verificou-se que quanto maior for Δf, maior é o tempo necessário para a malha entrar em sincronismo. Quando Δf é inferior à largura de banda da PLL (50Hz neste caso), o sincronismo é obtido quase instantaneamente.

Na Figura 2.10 estão representadas as mesmas situações observadas anteriormente, mas agora com um sinal de entrada adicionado de ruído.

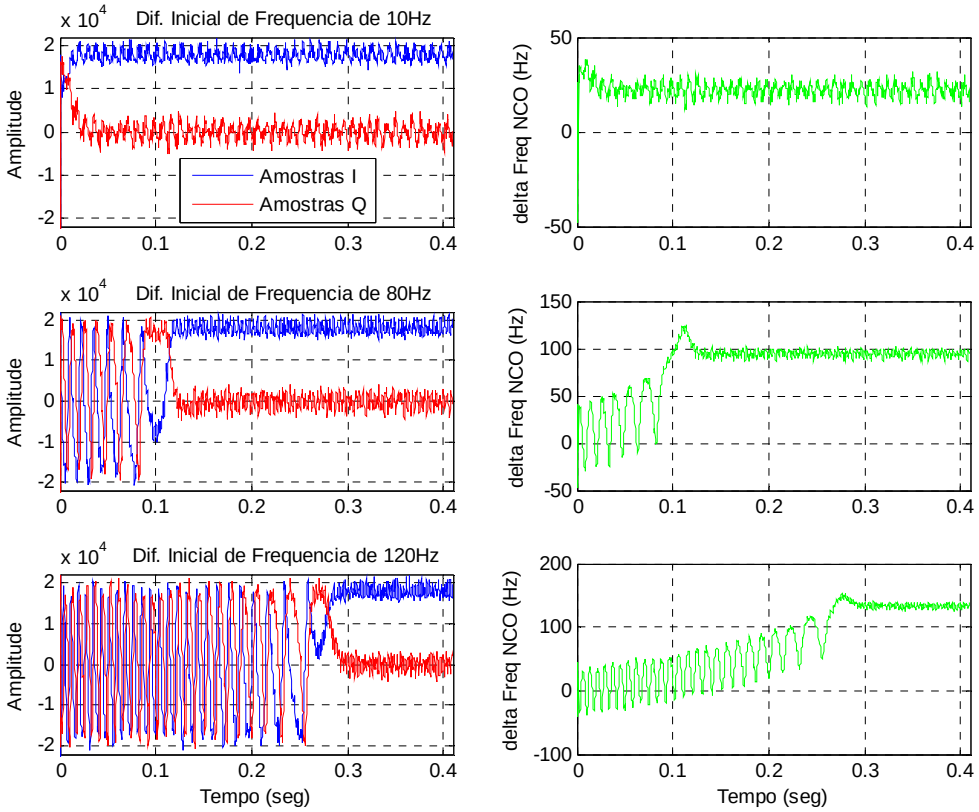

Figura 2.10 – Exemplos de aquisição de sincronismo na presença de ruído.

Como é óbvio, a variância da frequência e das componentes I e Q aumenta bastante na presença de ruído. Este aumento pode ter um grande impacto na aquisição e manutenção do sincronismo, especialmente quando a SNR do sinal de entrada se degrada. Nestes casos, a malha passa a necessitar de mais tempo para adquirir o sincronismo e mais facilmente o perde.

Como se referiu anteriormente nas secções 2.1.8 e 2.1.9, as perturbações introduzidas pelo ruído levantam assim muitas questões inerentes ao funcionamento do sistema, Torna-se então fundamental que o sistema seja adeaquado às condições favoráveis à aquisição e manutenção de sincronismo.

2.2. Controlo Automático de Ganho (AGC)

Geralmente a PLL é requerida para situações em que existem largas variações da amplitude do sinal de entrada, como acontece neste trabalho, já que o objectivo principal do sistema é precisamente o de medir a atenuação do sinal do satélite.

Todavia, de acordo com o que vem referido em [3, pág. 13], o ganho do detector de fase K_d é proporcional à amplitude A do sinal de entrada, pelo que o ganho da malha $K=K_oK_d$ é também proporcional a A. Similarmente, de acordo com as equações (2.15) e (2.16), os parâmetros do filtro, ou seja, a frequência natural ω_n e o coeficiente de

amortecimento ζ, também dependem de A. Esta limitação impossibilita a implementação da PLL para um desempenho óptimo, com um filtro com parâmetros fixos. Por outras palavras, se a amplitude do sinal de entrada varia, então também os parâmetros do filtro terão que variar de forma a manter a proporcionalidade com A. No caso de um sinal de entrada de baixa amplitude a malha será de banda estreita enquanto que se A for elevado tenderá a ser de banda larga.

Na implementação da PLL, é bastante importante considerar que o desempenho do sistema em presença de ruído é também fortemente afectado pela variação de A, através dos parâmetros da malha, K, ω_n e ζ. Uma das soluções possíveis para ultrapassar o problema da variação destes parâmetros com a amplitude do sinal de entrada, consiste na utilização de um sistema de controlo automático de ganho ou AGC (*Automatic Gain-Control*), que permitiria manter o comportamento dinâmico da malha e o seu desempenho no seguimento.

Os sistemas de controlo automático de ganho têm grande importância nos receptores de comunicação modernos e surgiram devido ao facto de que os sinais estão geralmente sujeitos a atenuação (*fading*), que se traduz por variações de amplitude na portadora[1]. Este tipo de circuito é usado em diversas aplicações que requerem a manutenção constante dos níveis do sinal de saída.

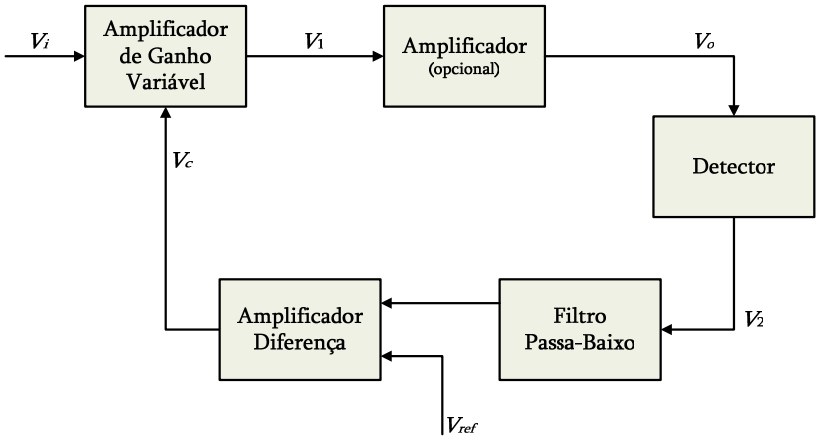

Figura 2.11 – Diagrama de blocos de um sistema de controlo automático de ganho ou AGC.

Na Figura 2.11 estão representados os elementos básicos de um AGC. O sinal de entrada V_i é amplificado por um amplificador de ganho variável, sendo que o sinal resultante V_1 pode ainda ser novamente amplificado para gerar o sinal de saída V_o. O papel do detector é seleccionar uma determinada característica V_2 do sinal de saída, como por exemplo a amplitude da portadora, que depois é filtrada num filtro passa-

[1] Os primeiros circuitos AGC eram denominados por *Automatic Volume-Control* e surgiram com o aparecimento dos primeiros aparelhos de rádio. A atenuação do sinal obrigava estes aparelhos a terem que ajustar continuamente o ganho do receptor, consoante a amplitude do sinal recebido, para que pudessem manter um sinal de saída constante, que neste caso era o volume. Mais tarde a utilização deste tipo de circuitos foi generalizada e passou a usar-se a denominação de *Automatic Gain-Control*.

baixo e comparada com um sinal de referência. Esta comparação é efectuada por um amplificador diferença para gerar o sinal V_c, responsável por controlar o amplificador de ganho variável. Este sinal de controlo é portanto o sinal de erro entre a o sinal V_2 e o sinal de referência V_{ref}. O AGC representado na Figura 2.11 é assim um sistema com *negative-feedback*, em que à medida que a amplitude do sinal de entrada varia, a tensão se controlo também varia de forma a minimizar o sinal de erro.

Na Figura 2.12 está representada a característica de um sistema AGC típico. Como se pode observar, quando a amplitude do sinal varia entre V_1 e V_2, o AGC mantém a amplitude do sinal de saída praticamente constante. No entanto, para sinais inferiores a V_1 ou superiores a V_2, o AGC é inoperante e o sinal de saída varia linearmente com o sinal de entrada.

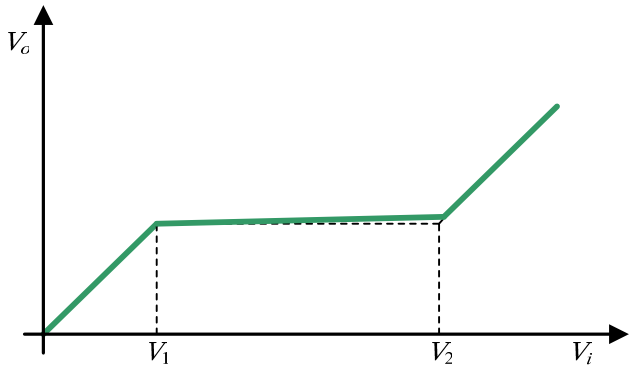

Figura 2.12 – Característica de um AGC típico.

Assim, esta a gama funcional do AGC constitui um parâmetro fundamental no projecto do próprio circuito, pelo que são necessários alguns cuidados. Por um lado, a gama funcional deve estar a um nível suficientemente elevado para evitar a degradação do desempenho na presença de ruído. Por outro lado, este nível do sinal não deve ser muito elevado de modo a que os limites de funcionamento dos vários elementos não sejam ultrapassados, evitando assim distorções.

2.2.1. Largura de Banda do AGC

A largura de banda de um *loop* AGC é uma questão fundamental pois esta define a sua velocidade de resposta. Se a amplitude do sinal de entrada tiver variações muito bruscas e a resposta do AGC for muito lenta, o ganho do receptor não é compensado tão rapidamente como devia, sendo de esperar uma redução no sinal à saída do AGC e consequentemente a malha pode perder o sincronismo. A largura de banda terá que ser suficiente para permitir uma resposta rápida às variações do sinal de entrada.

No entanto, quanto maior for a largura de banda do AGC, mais exposto estará a malha ao ruído, pelo que a flutuação do ganho do receptor com o ruído será obviamente maior. É de esperar que estas flutuações possam interferir adversamente no seguimento

da PLL, especialmente se a largura de banda do AGC e a largura de banda da malha de seguimento forem comparáveis.

2.3. FLL – *Frequency Locked Loop*

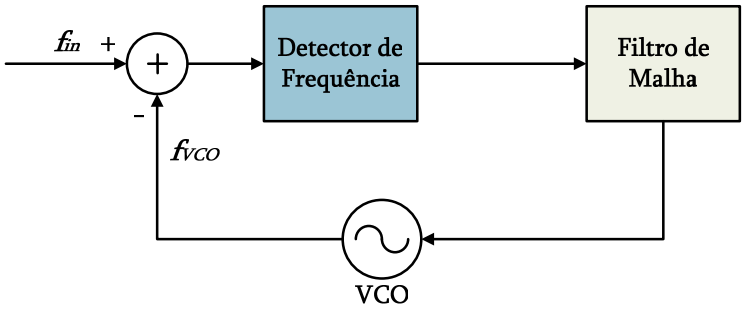

Figura 2.13 – Diagrama de blocos de uma FLL ou malha de seguimento de frequência.

A malha de seguimento de frequência é igualmente uma opção. Como se pode observar no diagrama de blocos apresentado na Figura 2.13, a diferença do ponto de vista funcional entre a malha da PLL e a FLL consiste na substituição do detector de fase por um detector de frequência. Em [4] são referidas vantagens sobre a utilização deste tipo de malha justificadas pela reduzida variância de frequência esperada de uma baliza de satélite e por outro lado ao facto de a PLL seguir a fase que se torna muito variável quando a relação sinal-ruido na malha se degrada.

A implementação da FLL foi parcialmente desenvolvida por uma estudante *Erasmus* [5], num trabalho apoiado por mim e pelo orientador. Consequentemente algumas partes são apenas apresentadas resumidamente.

Foi usado um detector de frequência do tipo correlação cruzada e implementadas duas malhas. Repare-se que enquanto o NCO numa malha de PLL funciona como integrador de fase, uma malha FLL com uma função de transferência similar à anteriormente descrita exige um duplo integrador. Um erro de frequência quase nulo é essencial para poder usar um filtro pré-detecção com uma reduzida largura de banda na detecção coerente.

2.3.1. Malhas Implementadas

O ganho do NCO corresponde à sua resolução de frequência, que tal como para a PLL é dado por:

$$K_o = \frac{f_{SAMP}}{2^{32}} = \frac{40\times10^6}{2^{32}} = 9.313\times10^{-3}, \tag{2.38}$$

onde f_{SAMP} representa a frequência do sinal de relógio do *chip* AD6620.

Por outro lado, de acordo com [5], o ganho do detector de frequência é dado por:

$$K_f = 2\pi A^2 T\,, \tag{2.39}$$

onde A e T representam respectivamente a amplitude das amostras e o período de amostragem.

Analisando a Figura 2.13, facilmente se conclui que se *F(s)* for a resposta em frequência do filtro de malha, então a resposta em frequência da FLL é dada por:

$$H(s) = \frac{K_f K_o F(s)}{1 + K_f K_o F(s)}\,. \tag{2.40}$$

Como não há uma relação linear entre o domínio S e Z, para obter as funções de transferência digital *H(z)* em analogia com as funções de transferência analógica *H(s)*, optou-se por recorrer à transformação linear, que permite obter muito boas aproximações desde que $s << T$.

A transformação linear é dada por:

$$z^{-1} = 1 - s \cdot T\,, \tag{2.41}$$

onde T representa o período de amostragem.

Em [5] foram implementadas duas configurações da FLL com dois filtros de malha distintos, nomeadamente um filtro de malha com um zero e um integrador e outro com um zero e dois integradores.

2.3.1.1. Malha com Um Pólo e Um Integrador

No caso do filtro com um integrador, é possível obter-se uma malha com uma largura de banda estreita e simultaneamente um erro de frequência reduzido. A resposta em frequência deste tipo de filtro é dada por:

$$F(s) = \frac{G}{s(s + \omega_p)}\,, \tag{2.42}$$

onde G é o ganho do amplificador DC.

Substituindo a equação anterior na equação (2.40), obtém-se a resposta em frequência da FLL digital:

$$H(s) = \frac{K_f K_o G}{s^2 + s\omega_p + K_f K_o G}\,, \tag{2.43}$$

que pode ser reescrita como:

$$H(s) = \frac{1}{\frac{s^2}{\omega_n^2} + \frac{2\zeta s}{\omega_n} + 1}\,, \tag{2.44}$$

onde a frequência natural e o coeficiente de amortecimento são respectivamente dados por:

$$\omega_n = \sqrt{K_f K_o G}\,, \tag{2.45}$$

$$\zeta = \frac{\omega_p}{2\omega_n}. \tag{2.46}$$

Nesta configuração, a frequência natural depende também da largura de banda da FLL (B_L):

$$\omega_n = 1.11\frac{2B_L}{\zeta + \frac{1}{4\zeta}}. \tag{2.47}$$

Substituindo a equação (2.41) na equação (2.42), obtém-se a resposta em frequência do filtro de malha com um pólo e um integrador:

$$F(z) = \frac{C_1}{C_3 z^{-2} - C_2 z^{-1} + 1}, \tag{2.48}$$

onde C_1, C_2 e C_3 são as constantes da malha dadas por:

$$C_1 = \frac{GT^2}{1+\omega_p T}, \quad C_2 = \frac{\omega_p T + 2}{1+\omega_p T}, \quad C_3 = \frac{1}{1+\omega_p T}. \tag{2.49}$$

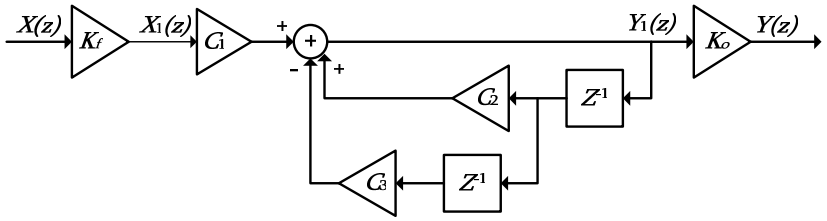

Figura 2.14 – Diagrama de blocos da FLL digital com um filtro de malha com um integrador.

Para determinar a resposta em frequência da FLL digital é necessário considerar também os efeitos do detector de frequência e do NCO. De acordo com o que está representado no diagrama de blocos da Figura 2.14, a resposta em frequência da FLL digital é então dada por:

$$H(z) = \frac{K_f K_o F(z)}{1 + K_f K_o F(z)} = \frac{K_f K_o C_1}{C_3 z^{-2} - C_2 z^{-1} + (1 + K_f K_o C_1)}. \tag{2.50}$$

Na Figura 2.15, está representado um gráfico com as respostas em frequência analógica e digital, considerando ζ = 0.707 (amortecimento crítico), B_L = 5Hz e T = 1/4882 e efectuando as correspondentes substituições nas equações descritas anteriormente.

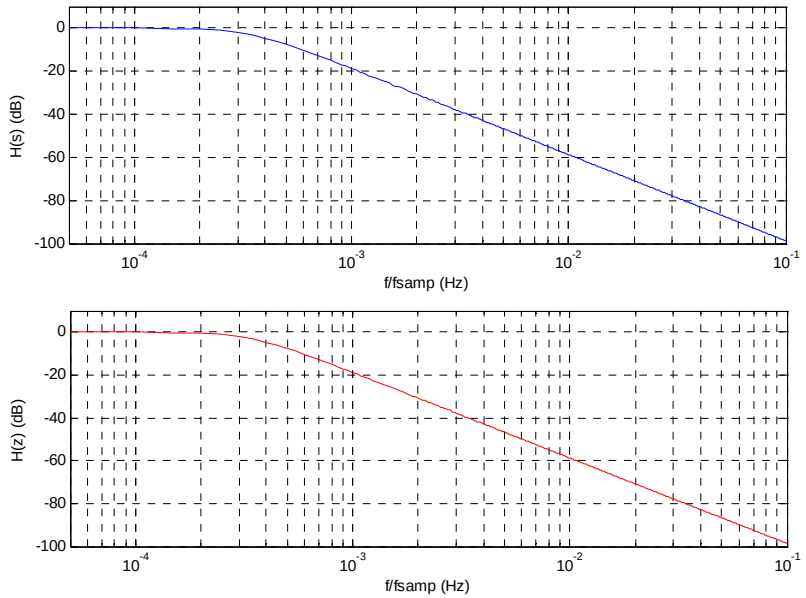

Figura 2.15 – Resposta em frequência da FLL analógica e da FLL digital com um integrador.

2.3.1.2. Malha com Um Zero e Duplo Integrador

A utilização de um filtro com um duplo integrador permite o seguimento com um erro de frequência nulo ao mesmo tempo que é possível também controlar a largura de banda, uma que esta não depende só do ganho da malha mas também dos parâmetros ω_z e G deste filtro. A resposta em frequência deste filtro é dada por:

$$F(s) = \frac{G(s+\omega_z)}{s^2}, \tag{2.51}$$

onde G é o ganho do amplificador DC.

Substituindo a equação anterior na equação (2.40), obtém-se a resposta em frequência da FLL digital:

$$H(s) = \frac{K_f K_o G s + K_f K_o G \omega_z}{s^2 + K_f K_o G s + K_f K_o G \omega_z}, \tag{2.52}$$

que pode ser reescrita como:

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{2.53}$$

onde a frequência natural e o coeficiente de amortecimento são respectivamente dados por:

$$\omega_n^2 = K_f K_o G \omega_z, \tag{2.54}$$

$$\zeta = \frac{\omega_n}{2\omega_z}. \tag{2.55}$$

Nesta configuração, analogamente ao que acontece na PLL e que está descrito na equação (2.23), a frequência natural depende também da largura de banda da FLL (B_L):

$$\omega_n = \frac{2B_L}{\zeta + \frac{1}{4\zeta}}. \tag{2.56}$$

Substituindo a equação (2.41) na equação (2.51), obtém-se a resposta em frequência do filtro de malha com um zero e um integrador:

$$F(z) = \frac{-C_2 z^{-1} + C_1}{z^{-2} - 2z^{-1} + 1}, \tag{2.57}$$

onde C_1 e C_2 são dadas por:

$$C_1 = GT + \omega_z T^2 G, \quad C_2 = GT. \tag{2.58}$$

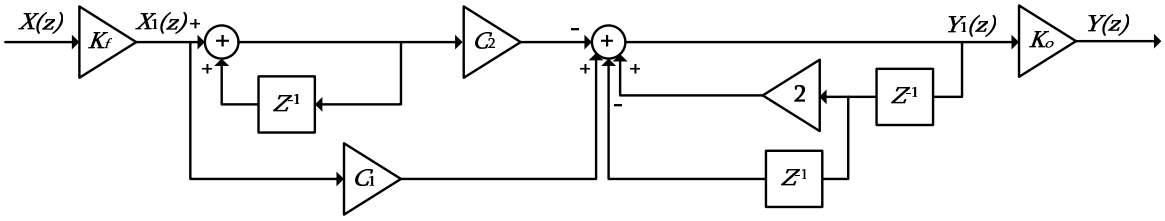

Figura 2.16 – Diagrama de blocos da FLL digital com um filtro de malha com um integrador.

Para determinar a resposta em frequência da FLL digital é necessário considerar também os efeitos do detector de frequência e do NCO. De acordo com o que está representado no diagrama de blocos da Figura 2.16, a resposta em frequência da FLL digital é então dada por:

$$H(z) = \frac{K_f K_o F(z)}{1 + K_f K_o F(z)} = \frac{-K_f K_o C_2 z^{-1} + K_f K_o C_1}{z^{-2} - (2 + K_f K_o C_2) z^{-1} + K_f K_o C_1 + 1}. \tag{2.59}$$

Na Figura 2.17, está representado um gráfico com as respostas em frequência analógica e digital, considerando ζ = 0.707 (amortecimento crítico), B_L = 5Hz e T = 1/4882S e efectuando as correspondentes substituições nas equações descritas anteriormente.

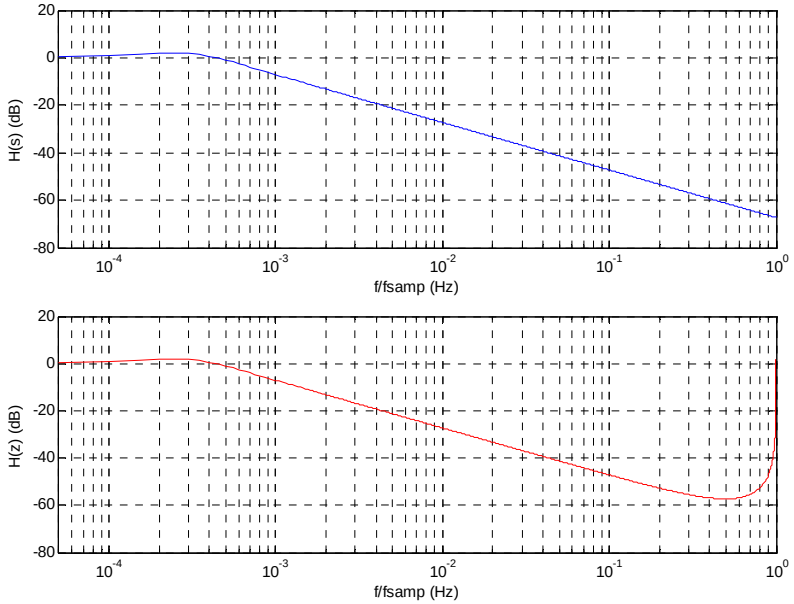

Figura 2.17 – Resposta em frequência da FLL analógica e da FLL digital com um integrador.

2.3.2. Resultados Experimentais

Os resultados que se apresentam de seguida foram obtidos utilizando a configuração com o filtro de malha com duplo integrador, considerando $\zeta = 0.707$, $B_L = 5\text{Hz}$ e $T = 1/4882\text{S}$. Neste caso, as constantes da malha C_1 e C_2 são dadas por:

$$C_1 = 6.6660 \times 10^{-7}. \tag{2.60}$$

$$C_2 = 6.6569 \times 10^{-7}. \tag{2.61}$$

sendo que a resposta em frequência é então dada por:

$$H(z) = \frac{-0.002731 z^{-1} + 0.002735}{z^{-2} - 2.002731 z^{-1} + 1.002735}. \tag{2.62}$$

Na Figura 2.18 estão representados alguns resultados obtidos com a FLL digital com a malha projectada anteriormente.

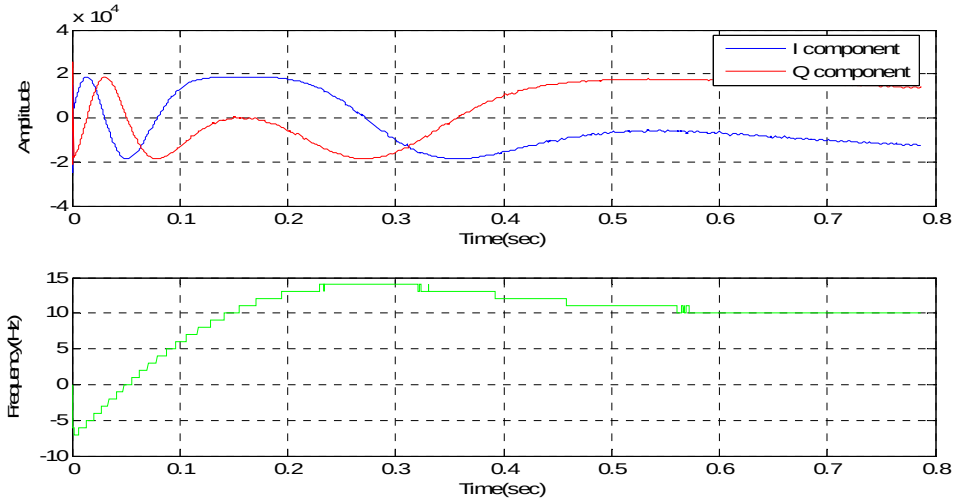

Figura 2.18 – Resultados obtidos com a FLL digital com B_L=5Hz e f_{in}=10.7MHz.

Alterou-se a largura de banda da malha para 10Hz. Neste caso as constantes da malha são dadas por:

$$C_1 = 1.33502 \times 10^{-6}. \tag{2.63}$$

$$C_2 = 1.33138 \times 10^{-6}. \tag{2.64}$$

sendo que a resposta em frequência é dada por:

$$H(z) = \frac{-0.005462 z^{-1} + 0.005477}{z^{-2} - 2.005462 z^{-1} + 1.005477}. \tag{2.65}$$

Os resultados obtidos com a nova malha estão representados na Figura 2.19.

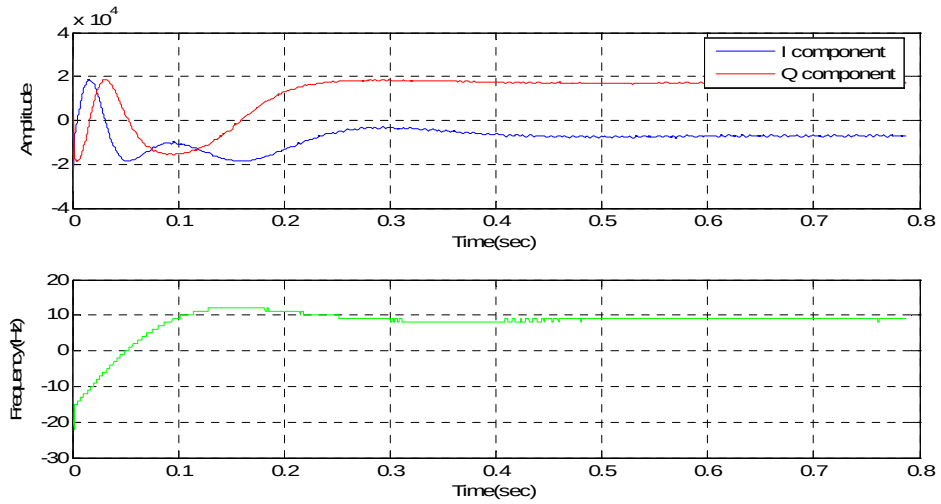

Figura 2.19 – Resultados obtidos com a FLL digital com B_L=10Hz e f_{in}=10.7MHz.

Comparando as duas figuras anteriores é possível observar que a malha adquire o sincronismo em qualquer dos casos. Contudo, como seria de esperar, quanto maior for a largura de banda da malha, mais rápida é a aquisição de sincronismo.

3. Arquitectura do *Software* do Sistema

As linhas gerais a conseguir pelo *software* desenvolvido consisiram em:

- Estimação inicial da frequência do sinal de entrada para o arranque do sistema;
- Implementação de um AGC para manter os parâmetros da malha constantes independentemente da atenuação do sinal;
- Avaliação das condições de funcionamento da malha e administrá-la consoante as necessidades;
- Disponibilização das componentes em fase e quadratura a uma taxa de pelo menos 10S/s para arquivamento em ficheiro.

Na Figura 3.1, está representado um diagrama de blocos que ilustra o funcionamento do sistema que se pretende implementar neste trabalho.

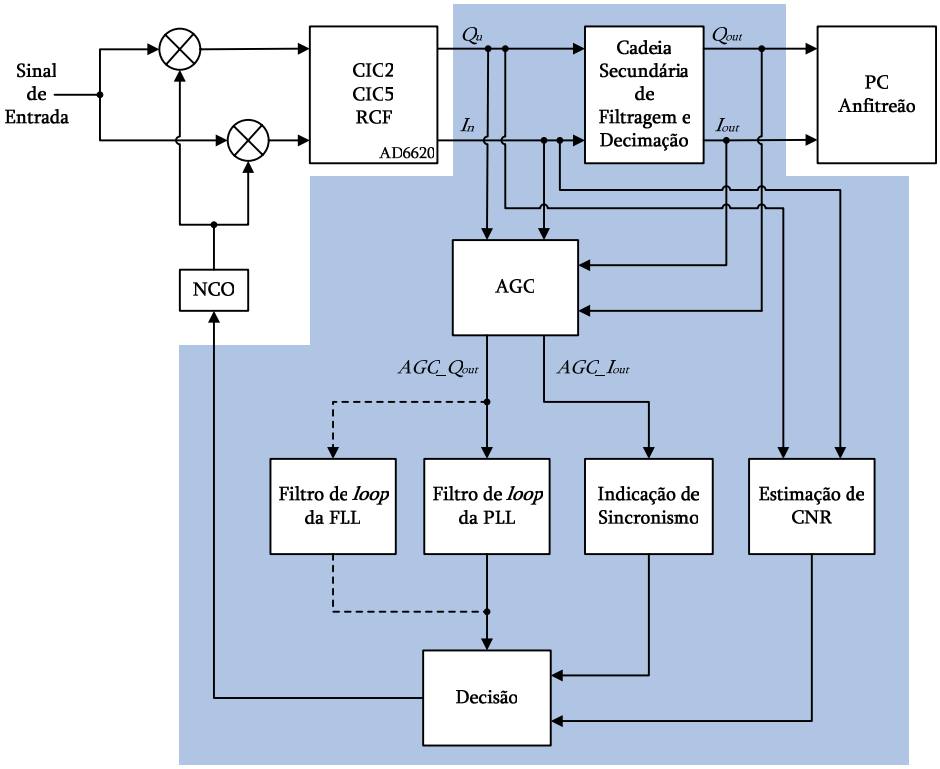

Figura 3.1 – Diagrama de blocos funcional do sistema a implementar.

Como se referiu no capítulo anterior, a diferença de frequência entre o sinal de entrada e o NCO tem que estar dentro de uma determinada gama para que PLL possa adquirir o sincronismo. Assim, antes de o sistema arrancar com a sua execução, torna-se fundamental estimar a frequência do sinal de entrada de forma a saber que frequência se deve configurar inicialmente no NCO. Por outro lado, este passo tem também que ser efectuado em situações após a perda prolongada de sincronismo. Na secção 3.1 é

abordada uma solução para esta questão, que se baseia na estimação espectral através de FFTs do sinal de entrada usando o NCO como oscilador de frequência variável.

Como se pode observar na Figura 3.1, a cadeia secundária de filtragem e decimação é responsável não só na definição da taxa de envio de dados em tempo real para o PC, como também pela definição da largura de banda do AGC. Estas implicações, fundamentais no que se refere ao desempenho global do sistema, são discutidas na secção 3.2.

Na secção 3.3 é abordada uma solução para a implementação do sistema de controlo automático de ganho ou AGC. Como se referiu no capítulo anterior, este sistema será utilizado para evitar os efeitos adversos da atenuação do sinal de entrada, mantendo a amplitude do sinal aplicado à malha da PLL essencialmente constante.

Depois de arrancar com a PLL digital, são necessários alguns blocos adicionais para gerir e controlar a sua execução, como são o caso da indicação de sincronismo e da estimação de CNR, que estão respectivamente descritos na secção 3.4.

De seguida, na secção 3.5 é abordado o dimensionamento da malha de seguimento de fase ou PLL digital.

O envio de dados em tempo real para o PC anfitreão é mandatório para os objectivos propostos para o sistema. Esta questão é abordada na secção 3.6, mas importa referir que apenas o protótipo de dois canais (ver capítulo 6) está pronto para executar esta tarefa, já que na construção do primeiro protótipo não foi projectada nenhuma solução viável para a transferência de dados em tempo real.

3.1. Estimação da Frequência do Sinal de Entrada

No domínio digital qualquer das técnicas de aquisição de sincronismo implementadas analogicamente e que estão descritas na secção 2.1.9.1, poderiam ser realizadas mas com certeza algumas seriam relativamente complexas de implementar.

Desta forma torna-se necessário recorrer a uma solução que, embora possa parecer algo semelhante às usadas numa solução analógica, é completamente distinta em termos de princípios funcionais. Esta solução baseia-se na estimação espectral por FFTs, onde o propósito seria encontrar um valor suficientemente próximo da frequência do sinal de entrada, de forma a permitir à PLL uma rápida aquisição de sincronismo. Para tal a frequência obtida terá que estar dentro da gama de *lock-in*, como se referiu no capítulo anterior.

Tipicamente a frequência do sinal de entrada poder-se-á encontrar numa janela de 20kHz em redor dos 10.7MHz. Este será com certeza um valor por excesso, já que a frequência da baliza de um satélite é muito estável e normalmente os piores valores apontados nas especificações são largamente sobrestimados.

Por exemplo, o satélite *HotBird-6*, actualmente a ser monitorizado na Universidade de Aveiro, foi observado durante um dia e o desvio de frequência não excedeu 100Hz contra os valores especificados de cerca de 2kHz. De qualquer forma desvios de frequência de longo termo (meses ou anos) poderão, se necessário, ser compensados pelos osciladores locais analógicos a montante do receptor digital.

Foi desenvolvido um novo protótipo com dois canais (ver capítulo 6), que inclui um circuito de síntese digital de frequência (DDS - *Direct Digital Synthesizer*). Pretende-se que o sinal sintetizado funcione como oscilador local, que poderá ser criteriosamente ajustado por *software* caso se detectem desvios de frequência significativos.

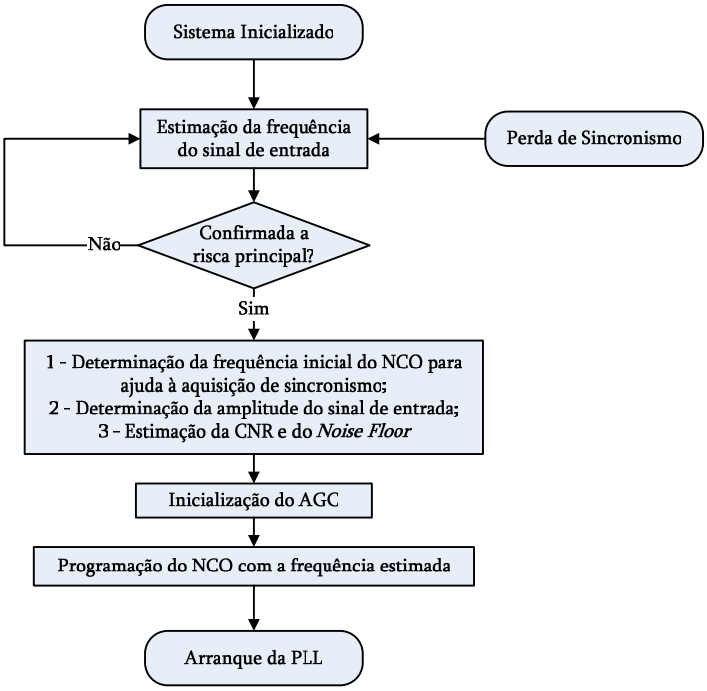

Figura 3.2 – Fluxograma do módulo de estimação da frequência do sinal de entrada.

Na Figura 3.2, está representando um diagrama de fluxo que mostra o funcionamento do módulo de estimação da frequência do sinal de entrada, de acordo com o que se pretende. Este módulo seria sempre executado em situações em que se desconhece a frequência do sinal de entrada, ou seja, no arranque, após a inicialização do sistema (configuração do AD6620 e das interfaces da DSP) ou após a perda de sincronismo devido a fortes atenuações do sinal.

Como mostra a Figura 3.3, o objectivo seria efectuar um varrimento no NCO com um incremento de frequência Δf, cobrindo toda a banda de variação do sinal.

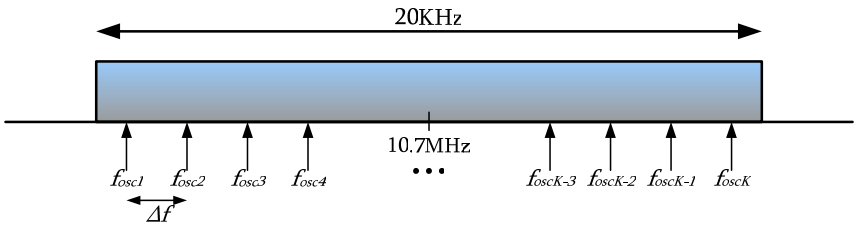

Figura 3.3 – Varrimento da banda do sinal utilizando o NCO como oscilador de frequência variável.

O passo inicial seria arrancar o sistema com a malha aberta e com o NCO programado com uma frequência correspondente a f_{osc1}. Seriam então recolhidas à saída do detector de fase, um número suficientede amostras para efectuar uma análise espectral por FFTs. Posteriormente repetir-se-ia este procedimento, depois de se incrementar a frequência do NCO em Δf para f_{osc2} e assim sucessivamente, até que toda a banda tenha sido varrida.

Detectando a risca principal entre os K espectros obtidos, seria possível estimar a frequência aproximada do sinal à saída do detector de fase, que corresponde à diferença de frequência entre o sinal de entrada e o NCO. Conhecendo a frequência central de varrimento para cada espectro, seria então possível estimar a frequência do sinal de entrada e logo o valor inicial para a frequência do NCO.

Todavia colocam-se algumas questões que devem ser exploradas antes de implementar esta solução:

- Algoritmo de análise espectral por FFT complexa;
- Resolução espectral;
- Discriminação da risca principal e estimação da densidade espectral de ruído;
- Incremento de frequência (Δf);
- Probabilidades de sucesso.

3.1.1. Algoritmo de Análise Espectral por FFT Complexa

Uma vez que esta técnica será aplicada apenas no arranque do sistema ou após a perda prolongada de sincronismo e, por outro lado, que a gama de *lock-in* garante uma aquisição ágil, tanto a rapidez como a precisão do algoritmo de análise espectral não constituem questões fulcrais. Desta forma, a escolha do algoritmo não é muito restritiva, pelo que inicialmente se optou por usar uma das funções para o cálculo de FFTs, disponíveis na biblioteca da DSP. No entanto, como se verá mais adiante na secção 5.1, não foi possível utilizar nenhuma das funções disponibilizadas por esta biblioteca, pelo que foi necessário recorrer a outra função.

Uma questão mais importante consiste em averiguar, em cada espectro obtido por varrimento, se a frequência estimada deve ser somada ou subtraída à frequência correspondente do NCO. A decisão será assim tomada consoante a localização da risca, uma vez que o espectro de um sinal complexo de frequência positiva apresenta a risca

principal na metade inferior, enquanto que num sinal de frequência negativa, esta está localizada na metade superior do espectro.

3.1.2. Resolução Espectral

Para cada análise espectral seriam adquiridas N amostras para uma resolução espectral, também conhecida como *resolution bandwidth*, dada por:

$$RBW = \frac{fa_{DF}}{N} \quad [\text{Hz/unidade}], \tag{3.1}$$

onde fa_{DF} representa a frequência de amostragem à saída do detector de fase.

Há no entanto algumas considerações importantes a ter em conta. Se a resolução espectral for muito fina, o número de amostras para cada análise espectral deve ser elevado, o que pode tornar algoritmo de varrimento demasiadamente pesado e moroso. Por outro lado, se a resolução espectral for muito grosseira, o número de amostras pode não permitir que a potência do sinal fique bem distribuída na largura de banda do sinal. Desta forma aumentaria também a potência média de ruído na largura de banda de resolução o que não é de todo desejável, especialmente se a potência do sinal fosse muito baixa.

Assim, na escolha do valor pretendido para a resolução espectral deverá existir um compromisso entre estas duas questões. Os valores de referência devem necessariamente ter em conta as características espectrais da risca da baliza. A densidade espectral de potência de uma baliza de satélite é tipicamente de -50dBc/Hz para afastamento da ordem de 10Hz. Desta forma pode-se assumir que em 20Hz de largura de banda está contida toda a potência do sinal. A variância de frequência por outro lado é de 20Hz mas os valores típicos observados são de cerca de 2 a 3Hz [6].

Neste trabalho os valores aceitáveis para a resolução espectral variam entre 10 e 50Hz.

3.1.3. Discriminação da Risca Principal e Estimação da Densidade Espectral de Potência de Ruído

Primeiramente é necessário analisar cada um dos espectros resultantes do varrimento de frequência no NCO e obtidos através de FFTs.

O primeiro passo seria determinar em cada espectro as riscas candidatas a risca principal. Esta última seria a risca cuja frequência estaria mais próxima da frequência das componentes I e Q à saída do detector de fase. No entanto, para conhecer estas riscas candidatas não bastaria saber qual a risca com maior amplitude em cada espectro (maior potência de sinal). Além disso seria necessário estimar a relação entre a potência de sinal e a potência de ruído em cada espectro. A risca principal seria então aquela que entre as riscas candidatas, apresentasse uma melhor relação de potências. Isto permitiria

ignorar as riscas que pudessem corresponder a picos de ruído (principalmente no caso do sinal ser fraco) e que poderiam ser facilmente confundidas com o sinal. Assim, apenas as riscas que apresentassem uma relação de potências superior a um determinado limite deveriam ser consideradas como válidas para estimação da frequência. Utilizando uma resolução espectral de cerca de 9.54Hz (512 pontos), foi considerado para candidatar uma risca o valor de referência de SNR=17dB (CNR-$10\log_{10}(9.54)$). Este é o valor que corresponde aproximadamente a uma CNR de 27dB/Hz (valor limite abaixo do qual se torna difícil a manutenção do sincronismo).

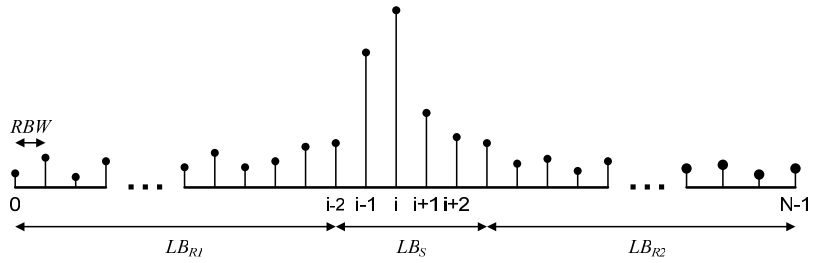

Figura 3.4 – Espectro obtido por análise espectral por FFT.

Um método para obter esta relação de potências em cada espectro consiste em:

- Estimar a potência do sinal através da soma das potências da risca de maior amplitude e das M riscas adjacentes:

$$P_S = \sum_{n=i-\frac{M}{2}}^{i+\frac{M}{2}} R_n^2 \quad [\mathrm{W}], \tag{3.2}$$

onde R_i representa a amplitude da risca candidata em cada espectro.

O número de riscas adjacentes M a considerar para a análise de cada espectro (que se pode observar na Figura 3.4), tem que ser o suficiente para cobrir a largura de banda do sinal (que variará entre 20 e 50Hz). Desta forma este número depende também da resolução espectral, já que esta define a largura de banda associada a cada risca. No caso desta variar entre 10 e 50Hz/amostra, o número de riscas adjacentes só poderá ser igual a 2, 4 ou 6, uma vez que estes são os números mínimos necessários para englobar uma largura de banda máxima de aproximadamente 50Hz.

- Estimar a potência do ruído através da densidade espectral de ruído (W/Hz), que por sua vez pode ser estimada a partir da potência média de ruído dada por:

$$P_{Rm} = \frac{\sum_{n=0}^{i-\frac{M}{2}-1} R_n^2 + \sum_{n=i+\frac{M}{2}+1}^{N-1} R_n^2}{LB_{R1} + LB_{R2}} \quad [\mathrm{W/Hz}], \tag{3.3}$$

onde o numerador corresponde ao somatório das potências de todas as riscas do espectro, exceptuando as riscas de maior amplitude e as adjacentes, e o denominador representa a largura de banda total correspondente a estas riscas.

No entanto, deve-se ter em conta que antes das amostras serem recebidas pela DSP, estas passam pelo último estágio de filtragem do *chip* AD6620 (filtro RCF). Assim, para o cálculo da potência média de ruído, deverão também ser excluídas as riscas fora da largura de banda de passagem do filtro RCF, ou seja, todas as riscas que correspondam a frequências superiores à frequência de corte deste filtro. Importa salientar que, para efeitos de simplificação, este facto não entrou em consideração na equação anterior nem nas equações que se seguem.

- Conhecida a potência média de ruído é estimada então a potência de ruído na largura de banda considerada para o sinal:

$$P_R = P_{Rm} \cdot LB_S \quad [\mathrm{W}], \tag{3.4}$$

onde LB_S representa a largura de banda que se considerou para o sinal.

Repare-se que a potência de ruído será formatada pelas respostas do filtro passa banda de *hardware* (que precede a ADC do sistema) e *software* pelo que poderá não representar com rigor a CNR nas vizinhanças da portadora contudo será uma boa estimativa. A redução do número de riscas do espectro poderia ser positiva neste aspecto, contudo a variância do valor anterior seria superior. Uma segunda estimativa mais rigorosa, mas seguramente sempre por defeito poderá ser efectuada após confirmar a risca principal e proceder ao ajuste do NCO para conseguir uma frequência de sinal próxima de zero. A compensação de desvios significativos de frequência deverá preferencialmente ser feita a montante do detector digital pela actuação num oscilador local que poderá ser a DDS do sistema de dois canais. De qualquer forma uma mensagem de aviso deverá ser transmitida caso se identifique a risca muito afastada do valor nominal ou a existência de mais que uma risca com SNR significativa pois são situações anormais.

- Calcular a relação entre as potências obtidas anteriormente:

$$SNR = \frac{P_S}{P_R}, \tag{3.5}$$

Depois de localizada a risca principal, poder-se-ia estimar a frequência correspondente efectuando uma média pesada das riscas à volta da que tem potência máxima. Desta forma, o valor obtido para esta frequência seria muito mais fiável do que o obtido recorrendo a uma única risca.

Este valor pesado pode ser obtido por:

$$i' = \frac{\sum_{n=i-\frac{M}{2}}^{i+\frac{M}{2}} \left(n \cdot R_n^2 \right)}{P_S}, \tag{3.6}$$

Todavia, há duas situações particulares que não foram até aqui consideradas e que correspondem aos casos em que as riscas de maior amplitude estejam localizadas muito perto ou mesmo nos extremos do espectro. Isto obrigaria a que as riscas a considerar para o sinal não fossem contíguas como se exemplifica na Figura 3.4Figura 3.3. Para seleccionar as riscas adjacentes nestes casos, deve-se considerar que a risca imediatamente anterior à risca de índice zero corresponde à última risca do espectro ($n = N$-1). Por outro lado a risca posterior à última risca do espectro corresponde à risca de índice zero. Por outras palavras, deve-se considerar que o espectro é circular.

As equações (3.2), (3.3) e (3.6) devem portanto ser adaptadas a estas situações, o que implica que:

- Se $n < 0$, então a amplitude desta risca é dada por: R_n, com $n = N+n$;
- Se $n > N$-1, então a amplitude desta risca é dada por: R_n, com $n = n-N$.

Como se verá mais adiante na secção 5.1, o cálculo da FFT discreta com um número finito de pontos origina o espalhamento de potência no espectro (*leakage*). O fenómeno de Gibbs é devido a uma observação limitada no tempo de um sinal. Como uma parte significativa da potência do sinal se distribui pelas riscas adjacentes à risca principal, este fenómeno pode levar à medição incorrecta das potências de ruído e de sinal e consequentemente da SNR.

Geralmente a forma mais utilizada para suprimir este fenómeno consiste na multiplicação do sinal por janelas (*windowing*). Existem várias funções janela como por exemplo a janela de Hamming, Gauss, Bartlett, Hann, etc. Embora os artefactos introduzidos não possam ser completamente suprimidos, esta técnica permite reduzir o espalhamento da potência do sinal pelo espectro levando contudo ao alargamento da risca.

Uma solução para obter uma estimativa da densidade espectral de potência de ruído seria através da potência média calculada por estimação espectral. No entanto, como foi referido o cálculo da potência de ruído pode vir afectado de erro devido ao *leakage*.

Para evitar ter que recorrer à utilização de janelas, uma opção mais simples passaria por efectuar alguns cálculos no domínio do tempo, usando apenas as amostras em fase e quadratura recolhidas à saída do detector de fase para a estimação espectral.

O primeiro passo seria calcular a média do sinal com ruído dada por:

$$\overline{x}_{SR} = \frac{1}{N}\sum_{n=0}^{N-1}\sqrt{I_n^2 + Q_n^2}\ , \tag{3.7}$$

onde N é o número de amostras recolhidas e I_n e Q_n são as amostras em fase e quadratura.

O passo seguinte seria calcular a média do ruído que pode ser obtida a partir da média do sinal com ruído:

$$\overline{x_R^2} = \frac{1}{N}\sum_{n=0}^{N-1}\left(\sqrt{I_n^2 + Q_n^2} - \overline{x}_{SR}\right)^2 . \tag{3.8}$$

Por fim, e dado que à saída do detector de fase o ruído é filtrado pelo filtro RCF, seria possível estimar a densidade espectral de potência de ruído dada por:

$$\eta = \frac{\overline{x_R^2}}{2f_{RCF}} \qquad \left[\mathrm{W^2/Hz}\right], \tag{3.9}$$

onde f_{RCF} é a frequência de corte do filtro RCF.

Seria também possível estimar o valor da CNR através da seguinte equação:

$$CNR = 10\log_{10}\left(2f_{RCF}\,\frac{\overline{x}_{SR}^2}{\overline{x_R^2}}\right) \qquad \left[\mathrm{dB/Hz}\right]. \tag{3.10}$$

Esta equação só dá uma boa estimativa quando a relação SNR é relativamente elevada, mas em qualquer dos casos espera-se sempre alguma variabilidade nas estimativas devido ao ruído.

3.1.4. Incremento de Frequência (Δf)

A escolha do valor para o incremento de frequência no varrimento depende da largura de banda de varrimento bem como da resolução espectral que se pretenda usar.

Quanto menor for o incremento de frequência, maior será o número de espectros necessários e vice-versa. Desta forma, se a resolução espectral for demasiado fina, isto pode impor um limite no incremento de frequência em prol da rapidez do algoritmo de varrimento. Por outras palavras, se o número de espectros a processar for elevado e além disso o tempo necessário para processar cada espectro for também elevado, então o processo de varrimento poderá ser muito mais lento do que o desejado.

Por outro lado, se o incremento de frequência for elevado, o número de espectros pode não ser o suficiente para cobrir toda a banda de varrimento, já que a largura de banda observada em cada espectro não é infinita. Como esta largura de banda corresponde à frequência de amostragem à saída do detector de fase, poder-se-ia pensar que este seria o limite superior para o incremento de frequência. No entanto, o *chip*

AD6620 apresenta no último estágio de filtragem o filtro RCF, cuja frequência de corte neste trabalho é largamente inferior à frequência de amostragem. Assim, o verdadeiro limite superior para o incremento de frequência, corresponde à largura de banda de passagem do filtro RCF. É por esta razão que na secção 3.1.3, se refere a necessidade de descartar todas as riscas dos espectros que estejam localizadas fora da largura de banda de passagem deste filtro.

3.1.5. Probabilidades de Sucesso

Para garantir elevadas probabilidades de sucesso, os compromissos descritos nas alíneas anteriores devem ser respeitados. Se mesmo assim os resultados forem duvidosos, o varrimento deve ser repetido as vezes que for necessário até que estes sejam fiáveis.

Para tal, importa que no final do varrimento se executem testes para averiguar a validade do valor de frequência obtido. Caso esta validade se verifique, o processo termina e o sistema pode arrancar com a frequência inicial do NCO estimada. Caso contrário, o processo repete-se até que o sucesso deste seja assegurado.

O aparecimento de duas riscas candidatas com SNR significativo configura uma situação anómala. A não detecção de qualquer sinal ao fim de algum tempo (da ordem de alguns minutos) igualmente configura uma anomalia no *hardware*, pois não são prováveis episódios de atenuação muito elevada e de longa duração. Como se verá mais adiante na secção 3.6, o que se propõe é que durante o processo de aquisição inicial, à semelhança do funcionamento de rotina, os dados sejam enviados em *frames* com mensagens de erro que deverão ser interpretadas pelo PC anfitrião.

3.2. Cadeia Secundária de Filtragem e Decimação

Como se referiu no início deste capítulo, a cadeia secundária de filtragem e decimação vai definir a largura de banda do AGC e também a taxa de envio de dados em tempo real para o PC anfitrião. Como os valores pretendidos para estes parâmetros seriam muito próximos, por uma questão de simplificação, optou-se por partilhar a saída da cadeia para ambos os módulos, já que caso contrário, teria que se adicionar um estágio adicional de filtragem e decimação.

Os filtros mais indicados para esta cadeia são filtros FIR (*Finite Impulse Response*), já que estes são inerentemente estáveis. Ao contrário do que acontece com os filtros IIR (*Infinite Impulse Response*), as amostras à saída dependem apenas das amostras do sinal de entrada, ou seja, não existe realimentação e logo também não há acumulação de erros de arredondamento. Os filtros IIR, apesar de permitirem um menor número de coeficientes (menor carga computacional) do que os filtros FIR, apresentam oscilações e podem-se tornar instáveis devido à realimentação.

No protótipo de um canal, a frequência de amostragem à saída do detector de fase é de aproximadamente 4882Hz, ao passo que a largura de banda da AGC deveria ser da ordem dos 20Hz. Para se obter uma taxa de amostragem desta ordem, seria necessária uma decimação com um factor bastante elevado. Optou-se então por utilizar um factor de decimação de 200, ao qual corresponde uma taxa de amostragem de cerca de 24.41S/s. Por outro lado, o filtro passa-baixo, que é obrigatório antes de se efectuar uma decimação, teria que apresentar um decaimento muito acentuado e logo, exigiria um elevado número de coeficientes.

Ao contrário do que acontece com a estimação da frequência do sinal de entrada, neste módulo a carga computacional constitui uma preocupação fulcral. Isto deve-se ao facto de que o processamento de cada amostra deve ocorrer no período entre o final da sua recepção e o início da recepção da amostra seguinte, de forma a evitar a perda de amostras. Todavia, uma vez que cada um destes períodos não é muito longo (cerca de 0.18ms), este processamento poderá ter que passar por optimização de *software* ou eventualmente, caso esta optimização não resulte, dividido em vários períodos. Esta questão é discutida na secção 4.9.

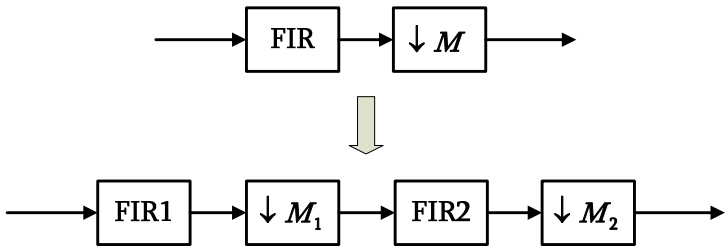

Figura 3.5 – Diagrama de blocos da cadeia secundária de filtragem e decimação.

Assim, uma melhor solução do ponto de vista de carga computacional, passa por dividir a filtragem e decimação em dois estágios, tal como se pode observar no diagrama de blocos da Figura 3.5, em que M_1 e M_2 representam os factores de decimação no primeiro e segundo estágio.

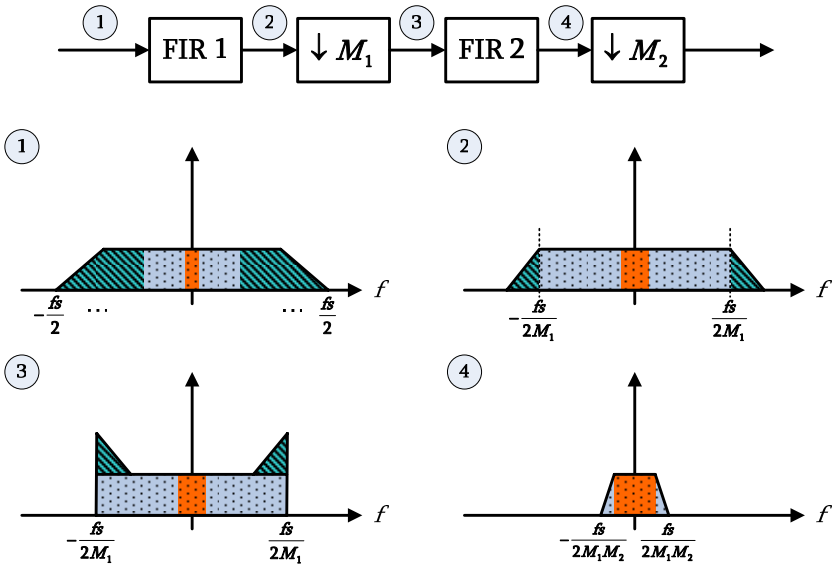

Figura 3.6 – Espectro do sinal ao longo da cadeia secundária de filtragem e decimação.

Na Figura 3.6 está representado o espectro do sinal ao longo da cadeia secundária de filtragem decimação, onde o sinal de interesse está representado a cor laranja. Esta é a banda que tem que ser preservada à saída da cadeia e que como se referiu anteriormente, corresponde à largura de banda do AGC e à taxa de envio de dados em tempo real para o PC. Assim, dividindo a filtragem e decimação em dois estágios, é possível reduzir a carga computacional ao reduzir a exigência de ambos os filtros mas principalmente do filtro FIR1, já que só é necessário preservar a banda de interesse.

Como se pode ver na Figura 3.6, é possível utilizar um filtro FIR1 com um decaimento mais reduzido. Mesmo que existam sobreposições de sinal após a decimação pelo factor M_1, as bandas onde estas ocorrem vão ser posteriormente filtradas no filtro FIR2. O que importa garantir é que a banda de interesse não seja afectada por estas sobreposições.

No projecto dos filtros FIR, existem ainda outros requisitos além dos referidos anteriormente. nomeadamente um *ripple* pequeno na banda de passagem e na banda de atenuação. O problema é que estes requisitos podem ser incompatíveis com uma reduzida carga computacional, na medida em que quanto mais exigentes são os filtros, maior será o número de coeficientes necessário. Assim, importa procurar um compromisso para que nenhuma destas exigências seja comprometida.

A solução encontrada para a implementação da cadeia de filtros decimadores está ilustrada na Figura 3.7 e baseia-se na utilização de *buffers* circulares para armazenar as amostras de entrada em cada estágio de filtragem e decimação. Estes *buffers* têm uma capacidade de memória igual ao número de coeficientes do respectivo filtro FIR. Neste tipo de *buffers*, as amostras vão sendo armazenadas para que as amostras mais recentes substituam as mais antigas.

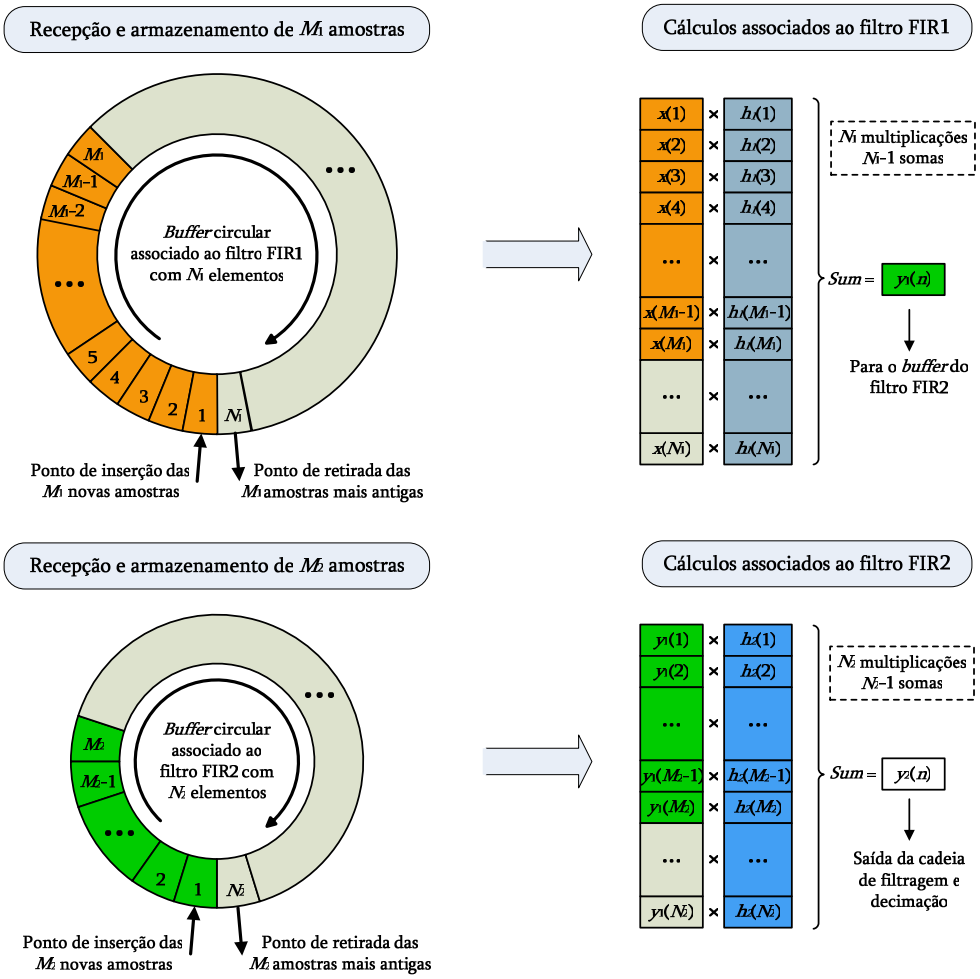

Figura 3.7 – Processamento das amostras na cadeia secundária de filtragem e decimação.

Como se pode observar, os cálculos associados ao filtro FIR1 são executados apenas a cada M_1 novas amostras recebidas, depois destas serem armazenadas no respectivo *buffer*. À medida que os valores à saída deste filtro vão sendo calculados, vão também sendo armazenados no *buffer* associado ao filtro FIR2. Por sua vez, os cálculos associados ao filtro FIR2 só são executados a cada M_2 novas amostras calculadas à saída do filtro FIR1. Por outras palavras, a cada M_{Total} novas amostras recebidas, é calculada uma única amostra à saída da cadeia secundária de filtragem e decimação.

Para projectar os filtros e garantir todos os requisitos necessários, recorreu-se a uma das muitas funções disponibilizadas pelo *Matlab*, neste caso a função *fipmord*. As linhas de comando executadas para esta simulação, incluindo o projecto dos filtros FIR, estão apresentadas na secção B.

Para que o sinal de interesse fosse preservado, o filtro FIR1 deveria ser projectado com uma frequência de corte superior a 12.21Hz ($f_s/2M_{Total}$) e com uma frequência de atenuação que não poderia ultrapassar os 109.85Hz (f_s/M_1-$f_s/2M_{Total}$), de modo a evitar o *aliasing* introduzido pela decimação de M_1. Quanto maior a diferença entre estes dois valores, menor será o decaimento e menor é o número de coeficientes do filtro. Por

outro lado, o filtro FIR2 deveria ser projectado com uma frequência de corte igual a 12.21Hz ($f_s/2M_{Total}$) e com uma frequência de atenuação inferior a 24.41Hz (f_s/M_{Total}) para evitar *aliasing*.

Optou-se então por projectar o filtro FIR1 com uma frequência de corte de 30Hz e com uma frequência de atenuação de 100Hz, enquanto que o filtro FIR2 foi projectado com uma frequência de corte de 12.21Hz e uma frequência de atenuação de 18Hz.

Na Figura 3.8 e na Figura 3.9 estão representados os gráficos das respostas em frequência dos filtros FIR projectados e cujos coeficientes se apresentam na Tabela 3.1 e na Tabela 3.2.

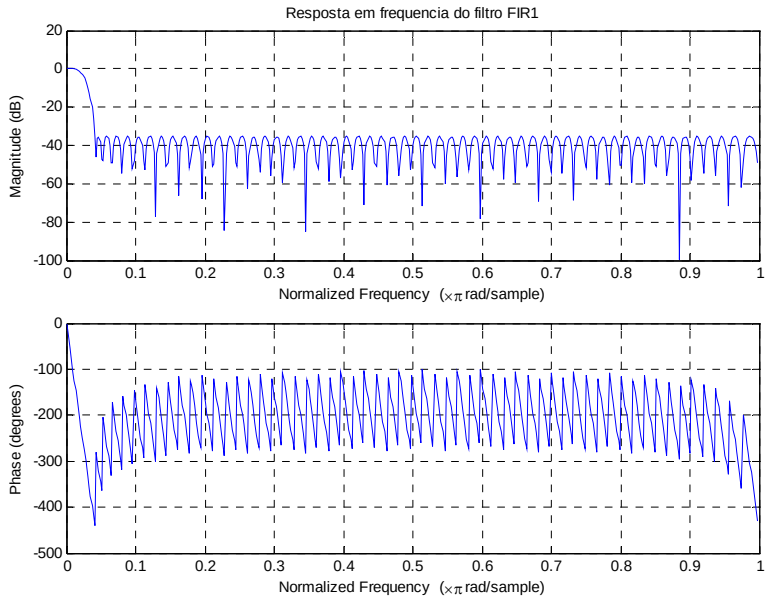

Figura 3.8 – Resposta em frequência do filtro FIR1.

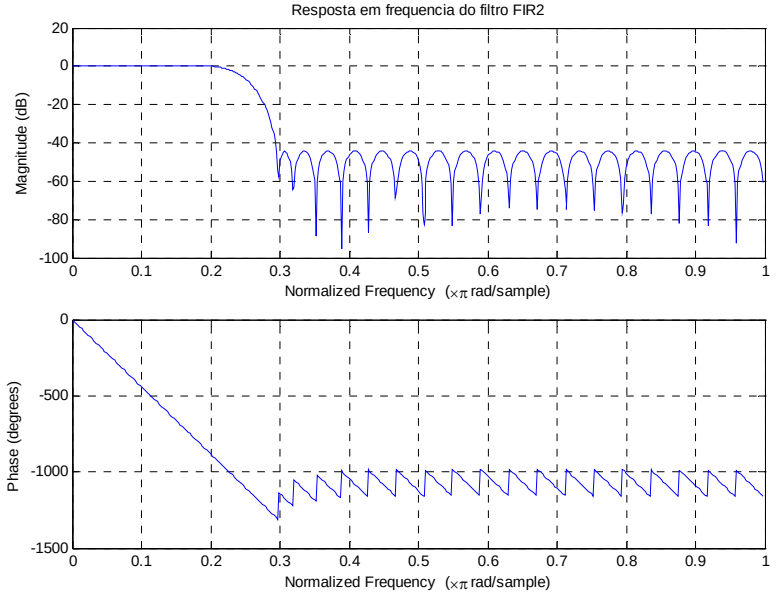

Figura 3.9 – Resposta em frequência do filtro FIR2.

n	$h_1(n)$	n	$h_1(n)$	n	$h_1(n)$	n	$h_1(n)$
0	-0.00873275	30	0.00503945	60	0.02758286	90	0.00394523
1	-0.00318910	31	0.00569221	61	0.02750404	91	0.00341506
2	-0.00092930	32	0.00682743	62	0.02733616	92	0.00236408
3	-0.00334291	33	0.00757758	63	0.02710169	93	0.00197283
4	-0.00143089	34	0.00874400	64	0.02676913	94	0.00095844
5	-0.00348937	35	0.00957796	65	0.02638689	95	0.00072816
6	-0.00185840	36	0.01075711	66	0.02589423	96	-0.00025087
7	-0.00359821	37	0.01165153	67	0.02537830	97	-0.00031221
8	-0.00218889	38	0.01283199	68	0.02473762	98	-0.00125830
9	-0.00364029	39	0.01376392	69	0.02409913	99	-0.00114886
10	-0.00240609	40	0.01492269	70	0.02332348	100	-0.00207490
11	-0.00359980	41	0.01586742	71	0.02258263	101	-0.00176821
12	-0.00250004	42	0.01698882	72	0.02168991	102	-0.00270868
13	-0.00345085	43	0.01791880	73	0.02086399	103	-0.00219518
14	-0.00243152	44	0.01898576	74	0.01987417	104	-0.00314909
15	-0.00314909	45	0.01987417	75	0.01898576	105	-0.00243152
16	-0.00219518	46	0.02086399	76	0.01791880	106	-0.00345085
17	-0.00270868	47	0.02168991	77	0.01698882	107	-0.00250004
18	-0.00176821	48	0.02258263	78	0.01586742	108	-0.00359980
19	-0.00207490	49	0.02332348	79	0.01492269	109	-0.00240609
20	-0.00114886	50	0.02409913	80	0.01376392	110	-0.00364029
21	-0.00125830	51	0.02473762	81	0.01283199	111	-0.00218889
22	-0.00031221	52	0.02537830	82	0.01165153	112	-0.00359821
23	-0.00025087	53	0.02589423	83	0.01075711	113	-0.00185840
24	0.00072816	54	0.02638689	84	0.00957796	114	-0.00348937
25	0.00095844	55	0.02676913	85	0.00874400	115	-0.00143089
26	0.00197283	56	0.02710169	86	0.00757758	116	-0.00334291
27	0.00236408	57	0.02733616	87	0.00682743	117	-0.00092930
28	0.00341506	58	0.02750404	88	0.00569221	118	-0.00318910
29	0.00394523	59	0.02758286	89	0.00503945	119	-0.00873275

Tabela 3.1 – Coeficientes do primeiro filtro FIR.

n	$h_2(n)$	n	$h_2(n)$	n	$h_2(n)$	n	$h_2(n)$
0	0.00193261	13	0.00923275	26	0.19524956	39	-0.01238624
1	-0.00273088	14	0.02209169	27	0.11747601	40	-0.00540253
2	-0.00372792	15	0.02415381	28	0.03615861	41	0.00254215
3	-0.00364318	16	0.01019697	29	-0.02380127	42	0.00704316
4	-0.00125950	17	-0.01617300	30	-0.04910643	43	0.00665799
5	0.00292872	18	-0.04187786	31	-0.04187786	44	0.00292872
6	0.00665799	19	-0.04910643	32	-0.01617300	45	-0.00125950
7	0.00704316	20	-0.02380127	33	0.01019697	46	-0.00364318
8	0.00254215	21	0.03615861	34	0.02415381	47	-0.00372792
9	-0.00540253	22	0.11747601	35	0.02209169	48	-0.00273088
10	-0.01238624	23	0.19524956	36	0.00923275	49	0.00193261
11	-0.01317167	24	0.24271833	37	-0.00510188		
12	-0.00510188	25	0.24271833	38	-0.01317167		

Tabela 3.2 – Coeficientes do segundo filtro FIR.

Poderia pensar-se que os cálculos para os filtros da cadeia secundária de filtragem e decimação seriam efectuados mais rapidamente em vírgula fixa do que em vírgula flutuante. No entanto as DSPs da série C6700 possuem *hardware* específico para executar instruções em vírgula flutuante, pelo que estas podem ser executadas tão rapidamente como as instruções em vírgula fixa (ver secção 4.9).

3.3. Controlo Automático de Ganho (AGC)

3.3.1. Largura de Banda do AGC

A largura de banda do AGC é condicionada pela dinâmica de variação de amplitude do sinal quando atenuado pela atmosfera. A atenuação devido a chuva é essencialmente um fenómeno lento associado a deslocamentos de células de chuva, sem contornos abruptos, e movendo-se com a velocidade do vento. A taxa de variação da atenuação só excepcionalmente excede os 0.2dB/s. O fenómeno de cintilação tem variações rápidas (até 5Hz) mas de qualquer forma de pequena amplitude.

Na Figura 3.10 está representado um diagrama de blocos do modelo de AGC que se pretende utilizar neste trabalho.

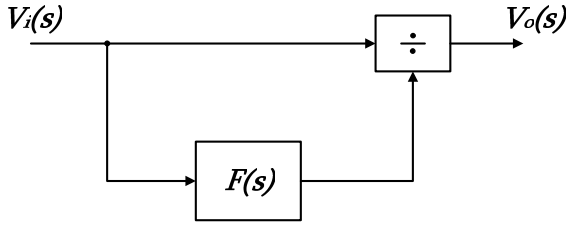

Figura 3.10 – Diagrama de blocos do AGC.

Se o filtro *F(s)* for um simples filtro passa-baixo de 1ªordem, a saída do AGC é dada por:

$$V_o(s) = \frac{V_i(s)}{V_i(s)F(s)} \Leftrightarrow V_o(s) = \frac{1}{F(s)} \Leftrightarrow V_o(s) = 1 + s\tau\,, \tag{3.11}$$

pelo que se pode concluir que, tal como se esperaria, a saída do AGC não depende da entrada. A equação (3.11) mostra que se a envolvente da amplitude da tensão tiver uma frequência elevada o AGC não reduz esta variação. De facto a saída do filtro passa-baixo será reduzida e não terá efeito no controlo de amplitude.

Esta largura de banda depende da largura de banda dos filtros passa-baixo presentes na cadeia secundária de filtragem e decimação. O seu valor corresponderá à frequência com que o AGC estima a amplitude do sinal à saída do detector de fase e, de acordo com o que se referiu na secção 2.2.1, deverá ser inferior à largura de banda da PLL.

3.3.2. Modelo de AGC

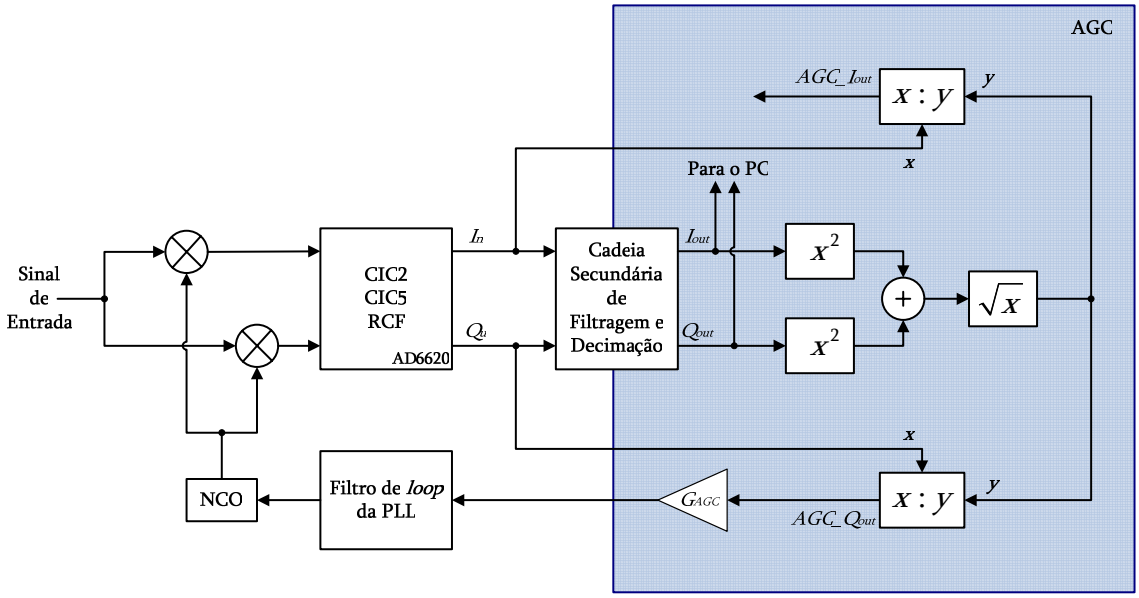

Figura 3.11 – Diagrama de blocos funcional do módulo de controlo automático do ganho.

Na Figura 3.11, está representado um diagrama de blocos que ilustra uma das possíveis soluções para o funcionamento do módulo de controlo automático do ganho. O princípio de funcionamento deste AGC baseia-se na detecção de amplitude do sinal de entrada. Analisando o diagrama pode-se constatar que no caso da PLL estar em sincronismo, é de esperar que se obtenham à saída do AGC dois valores praticamente constantes, rondando a unidade e o zero, dado que correspondem respectivamente à divisão das componentes I e Q, à saída do detector de fase, pela amplitude do sinal de entrada detectada. As saídas do AGC seriam então praticamente independentes da variação de amplitude do sinal de entrada. Assim, utilizando a saída AGC_Q_{out}, multiplicada por um valor de referência constante G_{AGC}, para fechar a malha da PLL digital, poderiam-se evitar os efeitos indesejáveis da variação da amplitude do sinal de entrada. Por outras palavras, seria possível projectar um filtro digital com parâmetros fixos e simultaneamente manter um desempenho quase óptimo.

Repare-se que a largura de banda de pre-detecção não deverá ser muito elevada caso contrário a estimativa do sinal terá um erro sistemático por excesso devido à detecção do ruído aditivo gaussiano. A indicação seria útil no sentido de diminuir a amplitude do sinal. Fica também a ideia que o AGC é ideal ou seja seria sempre obtido o valor AGC_I_{out}=1. Isto não é verdade porque a estimativa da amplitude do sinal efectuado na cadeia secundária de filtragem e decimação filtra também o ruído (diminui a sua potência e introduz atraso) das componentes catesianas originais. A função densidade de probabilidade desta variável foi simulada em *Matlab* para vários valores de atenuação assumindo o filtro RCF com 1kHz de largura de banda, CNR=55dBHz e a cadeia de estimativa da amplitude similar à apresentada anteriormente. Os resultados estão apresentados na Figura 3.12.

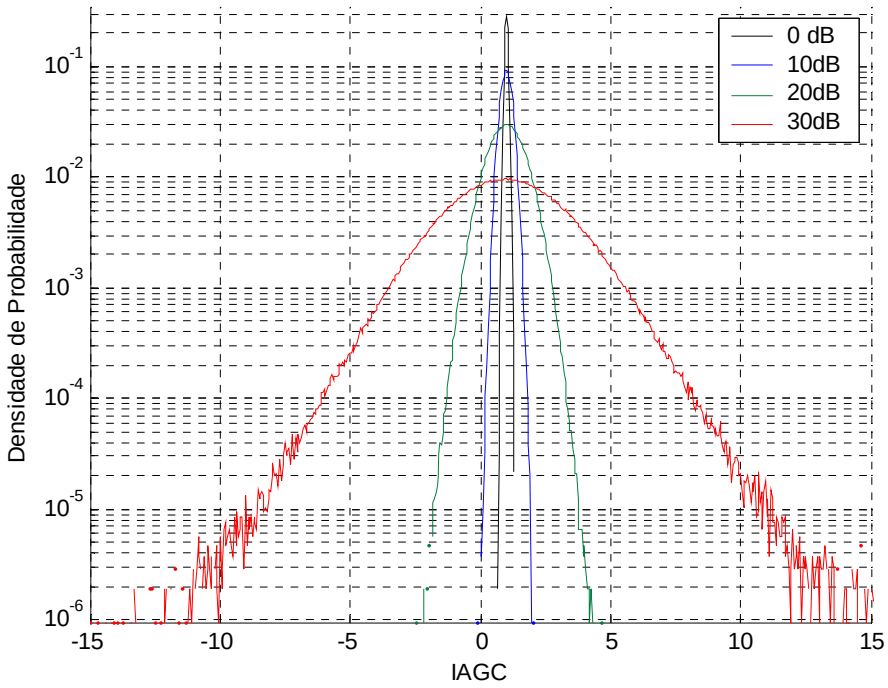

Figura 3.12 – Função densidade de probabilidade da variável AGC_I_{out}.

Como se pode observar o valor da variável assume valores elevados quando a potência de ruído domina a de sinal e valores muito próximos de 1 para elevados valores da CNR (atenuação nula). A média deste valor decresce com o aumento da atenuação, o que se designa por efeito de *signal supression*.

3.3.3. Simulação em *Matlab* do Modelo de AGC

Antes de passar à implementação em *software* do módulo de controlo automático de ganho, foi necessário primeiro efectuar uma simulação deste modelo de AGC. O objectivo seria averiguar a existência de atrasos causados pelos filtros potencialmente prejudiciais à exequibilidade do modelo.

Para tal recorreu-se ao *Simulink*, uma ferramenta do *Matlab* para simulação, modelação e análise de sistemas multidomínio dinâmicos.

Para garantir resultados fiáveis e conclusivos, a simulação deveria considerar uma situação mais próxima da situação real. Desta forma, além de se tentarem reproduzir as componentes I e Q acompanhadas de ruído, a simulação foi ainda realizada com os parâmetros muito próximos dos pretendidos para o sistema.

O modelo projectado com o *Simulink* está representado na Figura 3.13, em que os parâmetros utilizados foram os seguintes:

- Frequência de amostragem das componentes I e Q – 20S/s;
- Frequência de atenuação dos filtros passa-baixo – 10Hz;
- Frequência da onda modulante para simular ruído – 2Hz.

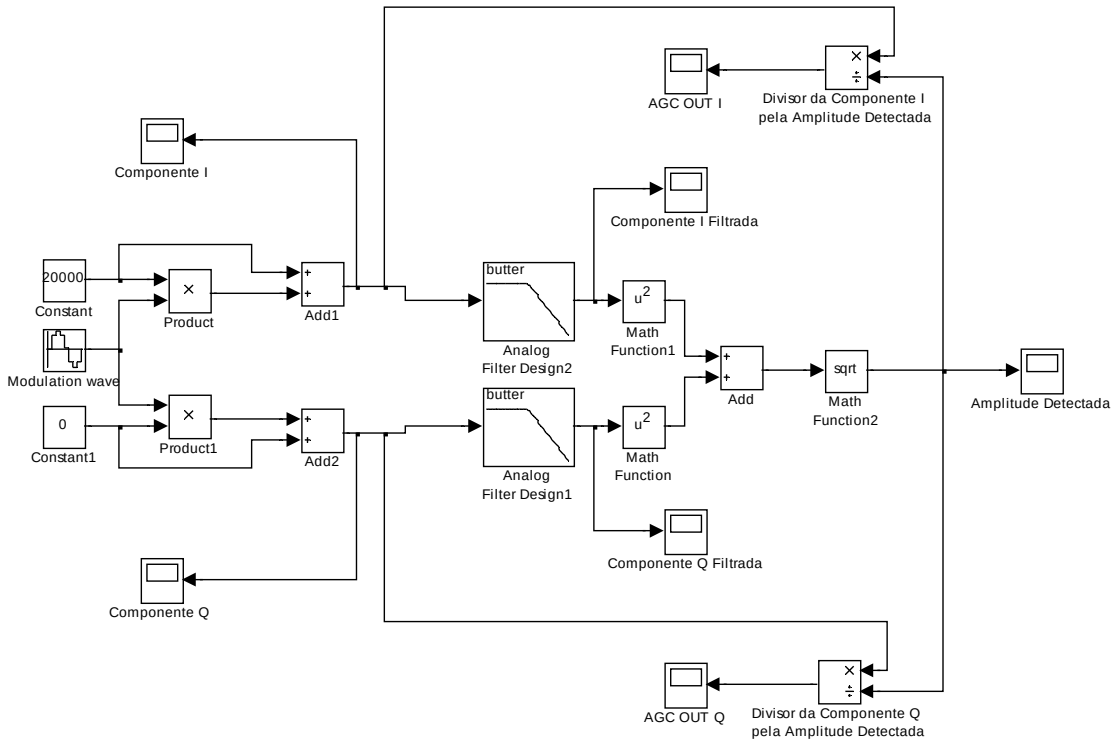

Figura 3.13 – Modelo projectado com o *Simulink* para simulação do sistema de AGC.

Os resultados da simulação estão representados na Figura 3.14, Figura 3.15, Figura 3.16 e Figura 3.17.

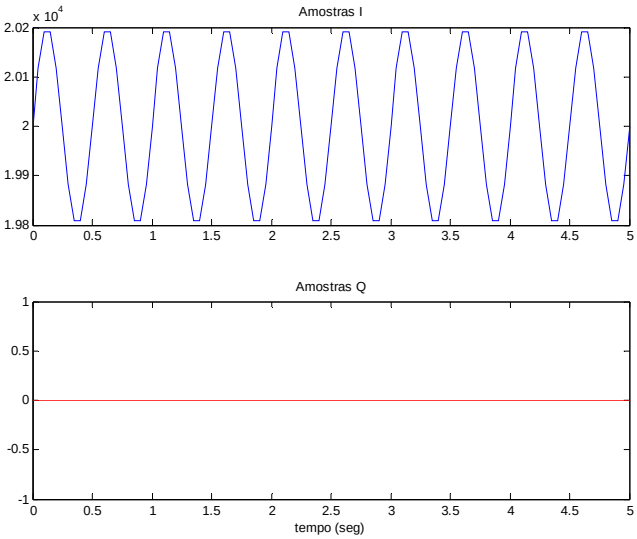

Figura 3.14 – Componentes I e Q simuladas (em sincronismo e moduladas).

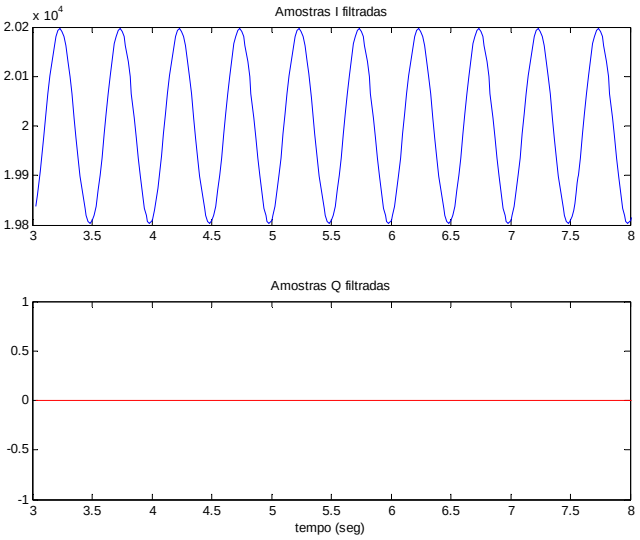

Figura 3.15 – Componentes I e Q filtradas.

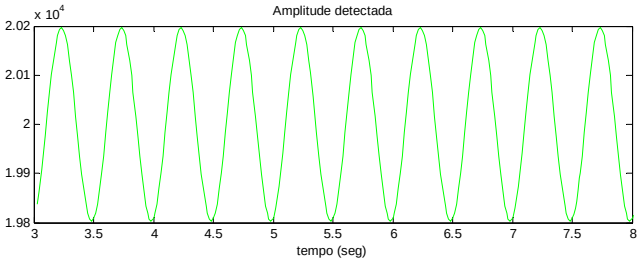

Figura 3.16 – Amplitude detectada.

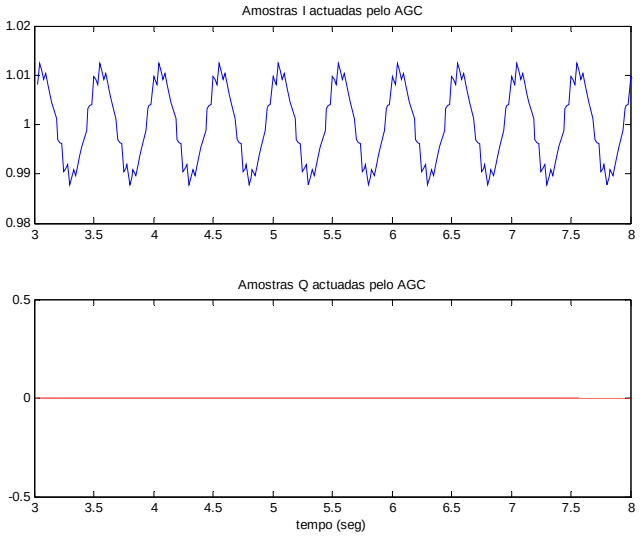

Figura 3.17 – Componentes I e Q à saída do AGC (*AGC_I*$_{out}$ e *AGC_Q*$_{out}$).

Como se pode observar pelas figuras anteriores, a componente I corresponde a um sinal modulado pela sinusoide modulante e a rondar os 20000, ao passo que a componente Q está a zero. As componentes mantêm o mesmo aspecto depois de filtradas, na medida em que a frequência do sinal modulante é inferior à frequência de

corte dos filtros passa-baixo. Uma vez que a componente Q está a zero, a amplitude detectada vai ser igual amplitude da componente I filtrada. Por fim, foi possível constantar o funcionamento do AGC de acordo com o que se pretendia. Apesar da amplitude do sinal ter uma variação considerável, a amplitude da componente I actuada pelo AGC mantém-se quase constante em redor do valor unitário (com uma variação muito reduzida).

3.4. Algoritmo de Administração da Malha

Na Figura 3.18 está representado um fluxograma que ilustra o algoritmo de administração baseado na avaliação constante das condições de funcionamento da malha, que é então gerida com o objectivo de manter o sincronismo ou recuperá-lo rapidamente.

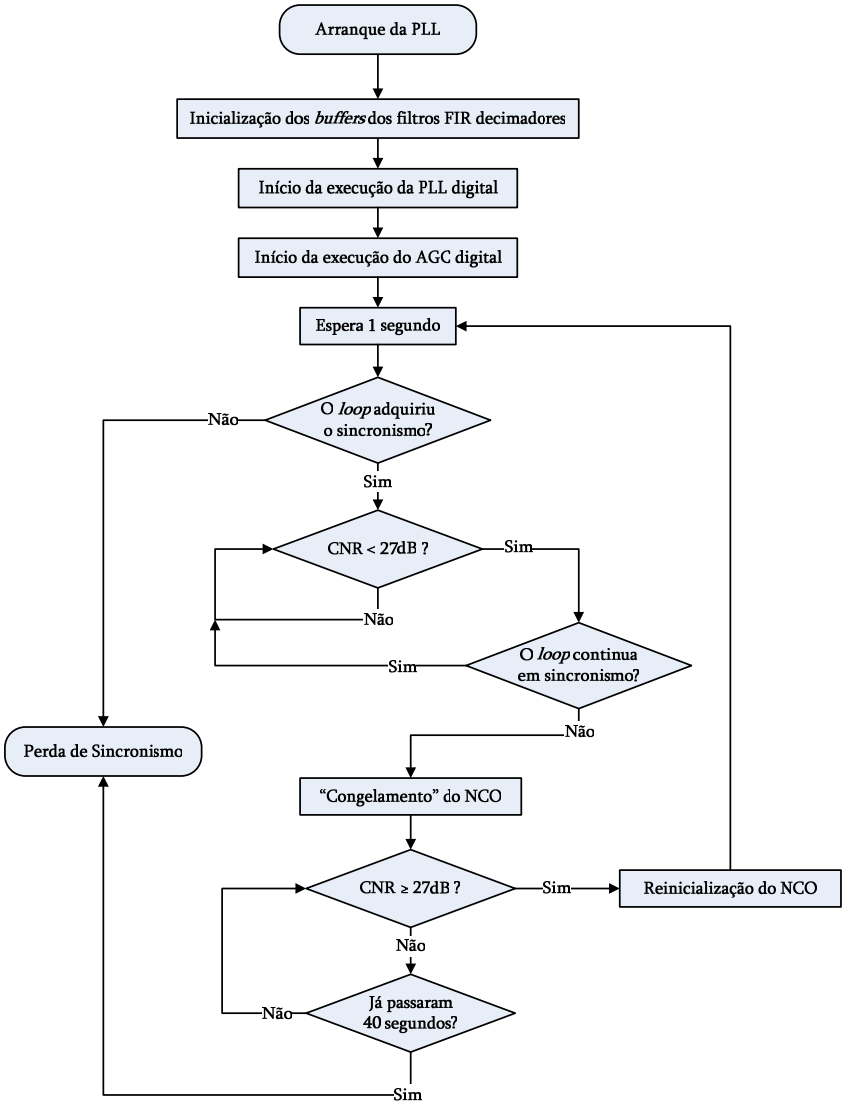

Figura 3.18 – Fluxograma do algoritmo de administração da malha.

3.4.1. Indicação de Sincronismo

Como se pode observar no fluxograma da Figura 3.18, depois do sistema conhecer o valor da frequência do sinal de entrada e arrancar com a execução da PLL digital, é necessário confirmar se a malha adquiriu o sincronismo. Caso isto não se verifique, o sistema deverá repetir a estimação da frequência do sinal de entrada e arrancar de novo a execução da PLL até que haja sucesso.

Como se referiu na secção 2.1.9.3, para se ter uma indicação fiável de sincronismo, é necessário efectuar a média da componente em fase e verificar se esta se encontra acima de um determinado valor de referência. Para se ter uma indicação de sincronismo a cada nova amostra recebida, em vez de utilizar um filtro passa-baixo (o qual implicaria um número de coeficientes provavelmente elevado, exigências de memória e alguma complexidade de administração) optou-se simplesmente por efectuar uma *running average* que é perfeitamente adequada às necessidades.

De acordo com [7, Cap. 32], uma *running average* consiste no cálculo de uma média aritmética de forma contínua e recursiva à medida que se vão recebendo os dados para os quais se pretende obter o valor médio, em vez de se calcular a média apenas depois de se terem recebido todos os dados.

Supondo que já se receberam n amostras e se obteve a respectiva média $\bar{x}_n$, pretende-se calcular a nova média $\bar{x}_{n+1}$, depois de recebida uma nova amostra. A forma mais directa para obter a média das n+1 amostras consiste em efectuar a soma de todas estas amostras e dividi-la posteriormente por n+1. No entanto, do ponto de vista computacional isto é obviamente inexequível, uma vez que tal implicaria a salvaguarda de todas as amostras recebidas. Por outro lado, caso esta solução fosse possível, também seria muito pouco eficiente na medida em que não se aproveitaria o facto da média $\bar{x}_n$ já ter sido calculada anteriormente.

Dado que é possível obter a soma das n amostras anteriores, multiplicando simplesmente a sua média $\bar{x}_n$ por n, a fórmula recursiva para a computação contínua da média aritmética, ou *running average*, das n+1 amostras é dada por:

$$\bar{x}_{n+1} = \frac{n \cdot \bar{x}_n + x}{n+1}, \tag{3.12}$$

onde x representa a última amostra recebida e $\bar{x}_0 = 0$.

Assim, em cada iteração é necessário guardar apenas o valor da média calculada e o número de amostras recebidas. No entanto, uma fórmula mais correcta para a *running average* em termos computacionais, já que poderá evitar possíveis *overflows* no cálculo da soma das amostras ($n \cdot \bar{x}_n$), consiste no desdobramento da equação anterior em duas fracções:

$$\bar{x}_{n+1} = \frac{n}{n+1}\bar{x}_n + \frac{x}{n+1}. \tag{3.13}$$

Além do cálculo da média, a outra questão importante reside na escolha do valor para o limite de decisão para a indicação de sincronismo, que como é óbvio, depende da amplitude do sinal de entrada, Assim, em vez de se efectuarem os cálculos da média sobre a componente I à saída do detector de fase, estes podem ser efectuados sobre a componente I à saída do AGC. Quando a malha está em sincronismo, a amplitude desta componente à saída do AGC é constante e ronda a unidade mas obviamente está sujeito às flutuações directas causadas pelo ruído na componente I e também pelo ruído na estimativa da amplitude do sinal no AGC. Desta forma, o limite de decisão poderia então ser um valor fixo, sem ter que variar com a amplitude do sinal de entrada. O valor escolhido foi de 0.9 o que, atendendo também ao longo tempo de integração, limita grandemente a probabilidade de falsas indicações de perda de sincronismo.

3.4.2. Estimação de CNR e "Congelamento" do NCO

Depois de confirmar que a malha está em sincronismo, é necessário estimar constantemente o valor de CNR do sinal de entrada, isto porque é necessário tomar algumas precauções em situações em que sinal se degrada ou degradou bastante, ou seja, em que a CNR atinge ou recupera de valores demasiado baixos. Nestes casos, a PLL estaria apenas a efectuar o seguimento com um sinal maioritariamente constituído por ruído, sendo que a frequência do NCO poderia sofrer desvios indesejados e consequentemente perder-se-ia o sincronismo.

Uma possível solução para esta questão passa por "congelar" o NCO assim que a CNR atinja um determinado valor mínimo, que deve ser de aproximadamente 27dB/Hz, como se pode observar no fluxograma da Figura 3.18. Este valor é tomado como um valor limite que conduz a um erro na estimativa da amplitude do sinal aceitável para a aplicação em causa. Contudo a PLL poderá manter-se em sincronismo em condições mais adversas e eisto será averiguado durante os testes. Este "congelamento" consiste basicamente na abertura da malha, cessando a execução da PLL digital. Após a abertura da malha, o sistema deve aguardar cerca de 40s até que o sinal recupere, ou seja, até que a CNR atinja aproximadamente os 27dB/Hz. Se o sinal recuperar dentro deste período de tempo, a malha deve ser novamente fechado mas agora efectuando o seguimento com um diferente valor de *offset* para a frequência do NCO. Este novo valor deve corresponder a uma média da frequência do NCO, que deverá ser efectuada durante um período de alguns segundos e que só deverá ser válida quando a CNR estiver acima de um determinado valor. Nestas situações, a malha deveria ser capaz de readquirir o sincronismo sem grandes problemas. Todavia, torna-se necessário averiguar se o sincronismo foi readquirido depois de reiniciar a execução da PLL. Para tal, após um tempo de espera de aproximadamente 1s, deve-se obter nova indicação de sincronismo. Se tal não se verificar, o sistema deve ser reinicializado e torna-se necessário repetir a estimação de frequência do sinal de entrada. O mesmo deverá acontecer se o sinal não recuperar durante o período de 40s, que é apenas um valor de teste plausível que estará relacionado com a brevidade de atenuações profundas do sinal.

Analogamente ao que se referiu anteriormente para a indicação de sincronismo, a média de frequência para definir o valor e também o *offset* de frequência do NCO bem como a estimativa de CNR irão ser calculadas com base numa *running average*. Particularmente a estimativa de CNR recorre a uma *running average* sobre a seguinte equação:

$$CNR = \frac{I_{out}^2 + Q_{out}^2}{\eta}. \tag{3.14}$$

onde I_{out} e Q_{out} são as componentes obtidas à saída da cadeia secundária de filtragem e decimação e η é a densidade espectral de potência de ruído obtida durante a estimação da frequência do sinal de entrada.

3.5. Dimensionamento da PLL Digital

Para dimensionar a PLL digital seguiram-se os mesmos passos que estão descritos em [3, págs. 11-14] e que podem também ser consultados em Anexos C. No entanto, os cálculos do filtro digital agora serão efectuados sobre a componente Q actuada pelo AGC e afectada pelo ganho G_{AGG} (de escolha arbitrária). Consequentemente, é o valor deste ganho que é utilizado para determinar o ganho do detector de fase, em vez da própria amplitude do sinal de entrada. Analogamente ao projecto anterior, assumiram-se valores para a largura de banda da PLL (B_L=50Hz) e para o coeficiente de amortecimento (ζ=0.707). A escolha deste último valor deve-se ao facto de se pretender um amortecimento crítico que será, à partida, mais indicado para o sistema do que o sub-amortecimento ($\zeta < 0.707$).

Optou-se por definir G_{AGG} =18500 (o valor nominal da amplitude do sinal com máxima amplitude) e efectuaram-se os referidos cálculos. De seguida apresentam-se os valores obtidos:

- K_d = 14.1752 unidades/rad;
- K_o = 9.3132×10^{-3} Hz/unidade;
- ω_n = 94.2856Hz;
- τ_1 = 14.8504μs;
- τ_2 = 14.9969ms;
- C_1 = 2.8244ms;
- C_2 = 0.2068s.

A resposta em frequência do filtro digital da PLL, que está representada na Figura 3.19, é aproximada por:

$$H(z) = \frac{-0.0274 z^{-1} + 0.0277}{z^{-2} - 2.0273 z^{-1} + 1.0277}. \tag{3.15}$$

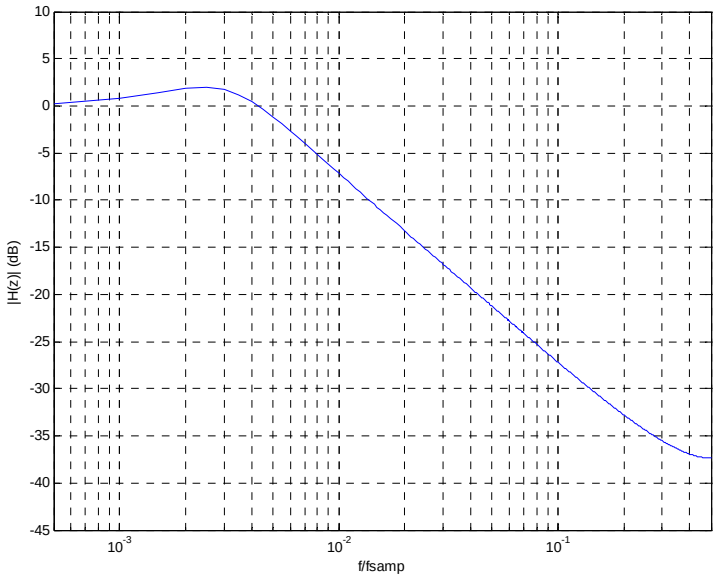

Figura 3.19 – Resposta em frequência da PLL digital – *H(z)*.

3.6. Envio de Dados em Tempo Real para o PC

Os principais dados necessários para a observação do sistema no PC são as amostras referentes às componentes I e Q, bem como os valores de frequência configurados no NCO ao longo do tempo. Para os efeitos de observação pretendidos, não é necessário enviar as amostras para o PC à mesma taxa com que são recebidas. Desta forma, os dados serão enviados a uma taxa igual à taxa de amostragem à saída da cadeia secundária de filtragem e decimação.

No entanto, tanto para efeitos de teste como para se ter uma ideia mais concreta do funcionamento do sistema, seria também interessante analisar os estados do sistema ao longo do tempo, bem como as estimativas de CNR e outras indicações. Uma solução possível para esta questão consistiria em enviar, além dos dados principais, uma palavra adicional com todas estas informações codificadas em *flags* binárias.

Depois de saber quais os dados a enviar para o PC, a questão fundamental reside na forma como a transferência pode ser efectuada, sabendo que o envio dos dados tem que ser feito rapidamente para não comprometer a recepção das amostras na DSP. Assim, a possibilidade de comunicação em tempo real utilizando a porta paralela esbarra desde logo na morosidade das suas operações de leitura e escrita, Os períodos destas operações podem ultrapassar largamente os tempos disponíveis entre a recepção de amostras consecutivas.

Optou-se então por recorrer ao único porto série McBSP disponível, apesar da sua utilização não ser tão directa como se possa pensar à primeira vista, já que estes portos não suportam o protocolo de comunicação série RS-232, utilizado pela porta série do PC. Assim de acordo com [8], além de implementar novo *hardware* para a conexão do

porto McBSP com a porta série do PC, tornou-se necessário implementar um novo módulo de *software* para simular uma comunicação série em conformidade com o protocolo RS-232. No entanto, uma vez que isto seria implementado em *software*, existe o receio da existência de um *overhead* que possa aumentar bastante o tempo de processamento entre a recepção de amostras consecutivas.

Importa ainda referir que estas alterações não foram efectuadas no protótipo de um canal, mas apenas no novo protótipo de dois canais, como se poderá ver mais adiante no capítulo 6.

4. Implementação do *Software*

O DSP *Starter Kit* (DSK) TMS320C6711 é fornecido com a ferramenta de *software* *Code Composer Studio*, que permite programar em linguagem C de alto nível. Esta ferramente utiliza a porta paralela existente no DSK para comunicação com o PC anfitreão. É através dela que os programas executáveis ("*.out*") para a memória interna da DSP e que são posteriormente executados. Por outro lado, é também por esta porta que se recebem ou enviam os dados de I/O a processar ou processados pela DSP.

Neste capítulo, descreve-se a estrutura do programa criado para o projecto, bem como o código implementado para os diversos módulos de *software* que o compõem, nomeadamente a estimação da frequência do sinal de entrada, a cadeia secundária de filtragem e decimação, a controlo automático de ganho (AGC), a indicação de sincronismo, a estimação de CNR e "congelamento" do NCO, a PLL e a FLL digital.

A rapidez de execução de todo o processamento requerido é uma exigência já que o processamento de cada amostra tem que ser terminado antes da recepção da amostra seguinte. Assim, como se poderá ver mais adiante na secção 5.2, foi necessário recorrer à optimização de *software* para reduzir o tempo de processamento. Na secção 4.9, é feita uma breve descrição de algumas das ferramentas mais importantes de optimização de *software* disponibilizadas pela DSP.

4.1. Estrutura do Programa

No código fonte"*conf.c*" foi necessário incluir alguns ficheiros de cabeçalho que contêm informação necessária para o desenvolvimento de *software* para os vários módulos da DSP, como declarações de constantes, declarações de variáveis, objectos e estruturas, protótipos e códigos de funções, macros, etc.

Foram incluídos os seguintes ficheiros de cabeçalho:

- *c6x.h* – biblioteca de suporte às DSPs da plataforma C6000;
- *csl.h* – biblioteca de suporte ao *chip* (CSL);
- *stdio.h* – biblioteca de suporte ao *standard* I/O;
- *stdlib.h* – biblioteca com as definições *standard*;
- *std.h* – biblioteca *standard* de C;
- *csl_emif.h* – biblioteca de suporte à EMIF;
- *csl_mcbsp.h* – biblioteca de suporte ao McBSP;
- *csl_irq.h* – biblioteca de suporte às interrupções;
- *regs.h* – biblioteca de suporte à manipulação de registos;
- *hwi.h* – biblioteca de suporte ao módulo HWI da API DSP/BIOS;
- *csl_legacy.h* – biblioteca que converte os nomes antigos para os nomes recentes;

- *math.h* – biblioteca de suporte a operações matemáticas;
- *conf.h* – biblioteca de suporte ao AD6620 e outros.

O ficheiro de cabeçalho *conf.h* que foi desenvolvido para este projecto, contém todas as definições de constantes, variáveis globais, estruturas e protótipos de funções que foram necessários para o desenvolvimento dos diversos módulos de *software*.

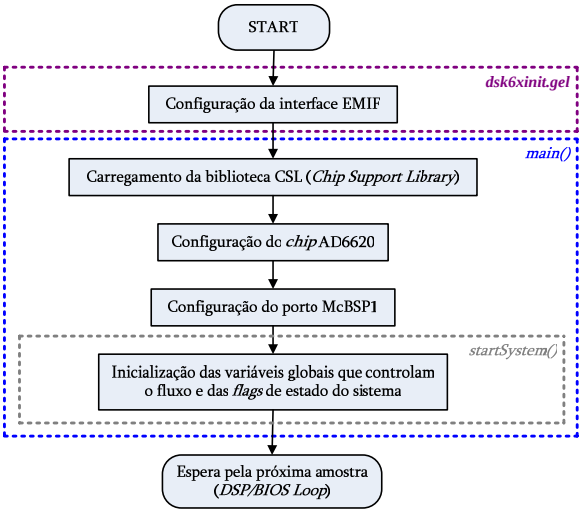

Figura 4.1 – Fluxograma de *software* da rotina principal *main()*.

Como se pode observar no fluxograma de *software* da Figura 4.1, as rotinas necessárias à inicialização do sistema são invocadas na rotina principal *main()*, que começa por carregar a biblioteca CSL (*Chip Support Library*), seguindo-se a configuração do *chip* AD6620 e a configuração do porto McBSP1. De referir ainda que a interface EMIF é configurada durante o *startup* pelo módulo *dsk6xinit.gel*. A partir de então a DSP entra no *loop* da DSP/BIOS, aguardando a ocorrência de uma interrupção RINT1, ou seja, a recepção de uma nova amostra.

O funcionamento básico do programa, que está representado no fluxograma de *software* da Figura 4.2, fica então a cargo da rotina de serviço à interrupção *RSI_RINT1()*, que é automaticamente evocada sempre que é recebida uma nova amostra e a DSP pode então receber e processar as amostras provenientes do AD6620.

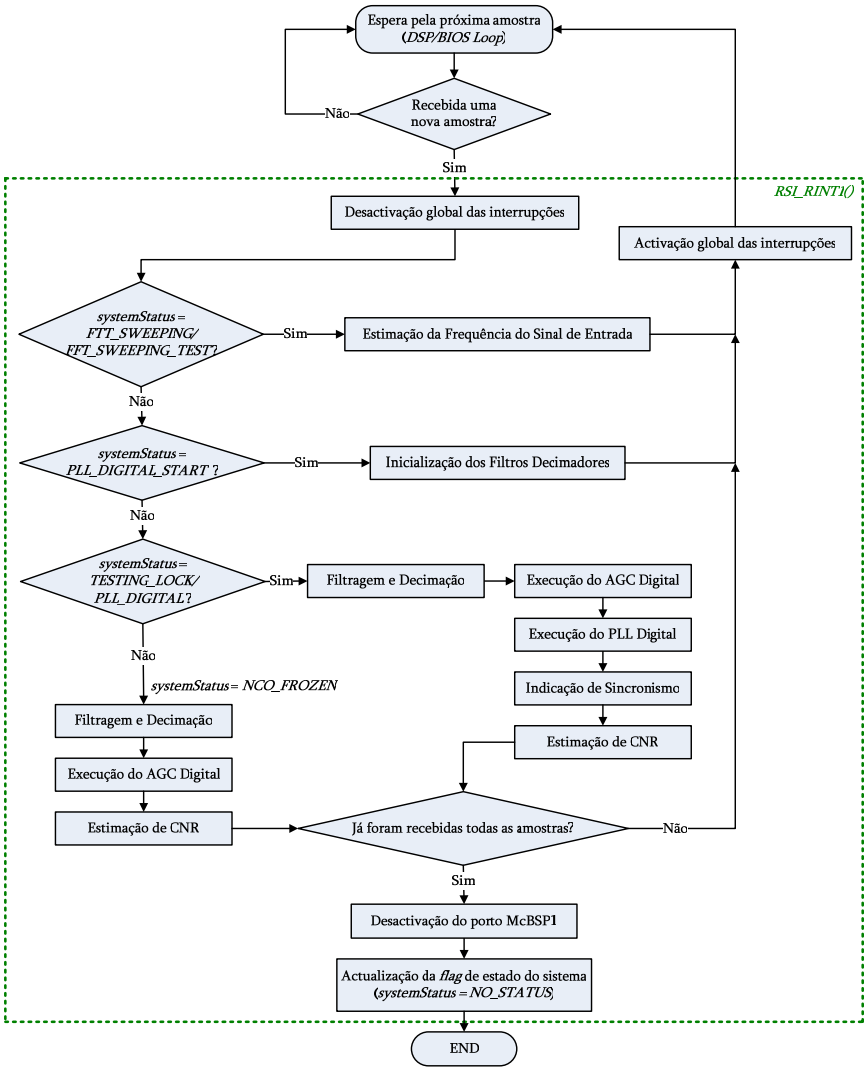

Figura 4.2 – Fluxograma de *software* do *loop* da DSP/BIOS e do rotina *RSI_RINT1()*.

De seguida apresentam-se as constantes, variáveis e rotinas mais importantes que foram definidas para utilização geral, ou seja, para utilização nos diversos módulos implementados neste projecto:

- Constantes:
  - *FREQ_CLK* = 40000000 – valor da frequência do relógio do *chip* AD6620;
  - *VAL_2_32* = 4294967296 – valor para a conversão a 32 bits (2^{32});
  - *N_AMOSTRAS* = 250000 – valor máximo do índice global de amostras;
  - *MAX_AMOSTRAS_INDEX2* = 1 – valor máximo do segundo índice global de amostras;
  - *N_AMOSTRAS_1SEG* = 5000 – número de amostras múltiplas de *MTOTAL* (ver secção 4.3) que correspondam aproximadamente a 1 segundo;
  - *NO_STATUS* = 0 – estado do sistema quando se encontra parado;

▫ *FFT_SWEEPING* = 1 – estado do sistema enquanto procede ao varrimento de frequência no NCO para estimar a frequência do sinal de entrada;

▫ *FFT_SWEEPING_TEST* = 2 – estado do sistema enquanto procede ao teste da frequência estimada para o sinal de entrada;

▫ *PLL_DIGITAL_START* = 3 – estado do sistema enquanto inicializa a PLL digital;

▫ *PLL_DIGITAL* = 4 – estado do sistema enquanto executa a PLL digital;

▫ *TESTING_LOCK* = 5 – estado do sistema enquanto testa o sincronismo;

▫ *NCO_FROZEN* = 6 – estado do sistema enquanto o NCO está "congelado".

- Variáveis globais:

  ▫ *unsigned int amostraIndex* – índice global das amostras a serem adquiridas e processadas;

  ▫ *unsigned int amostraIndex2* – segundo índice global das amostras a serem adquiridas e processadas;

  ▫ *short amostraI* – variável para armazenar a amostra I recebida;

  ▫ *short amostraQ* – variável para armazenar a amostra Q recebida;

  ▫ *unsigned char systemStatus* – *flag* do estado actual do sistema;

  ▫ *double freqNCO* – valor da frequência do NCO ou da saída do filtro digital da PLL/FLL.

- Variáveis locais:

  ▫ *int amostra* – variável local da rotina *RSI_RINT1()* para armazenar a amostra de 32 bits recebida pela porta série.

- Rotinas:

  ▫ *void RSI_RINT1(void)* - rotina de atendimento a interrupção (RSI) associada ao evento de recepção no porto McBSP1 - RINT1. O processamento das amostras recebidas é feito de acordo com a *flag* de estado do sistema *systemStatus*;

  ▫ *void startSystem(void)* – rotina que inicializa as variáveis globais que controlam o fluxo do sistema e as *flags* de estado do sistema.

As variáveis que controlam o fluxo do sistema são os contadores de amostras *amostrasIndex* e *amostrasIndex2*. O segundo índice define o tempo de execução do sistema, ou seja, o número de vezes que o primeiro é resetado a zero. O sistema termina a sua execução quando o índice *amostrasIndex2* atinge o valor da constante

MAX_AMOSTRAS_INDEX2. Para efeitos de simplificação, a actualização destas variáveis não está representada no fluxograma da Figura 4.2.

Aqui apenas se descreveram as rotinas e constantes mais importantes neste trabalho. As restantes rotinas e constantes, que dizem respeito à configuração das interfaces e do *chip* AD6620, estão descritas em [3, págs. 52-70].

4.2. Estimação da Frequência do Sinal de Entrada

De seguida apresentam-se as constantes, estruturas, variáveis e rotinas mais importantes que foram definidas na implementação do módulo de estimação da frequência do sinal de entrada:

- Constantes:
  - *N_ESPECTROS* = 5 – nº de espectros calculados por FFT no varrimento;
  - *N_AMOSTRAS_FFT* = 512 – nº de amostras para o cálculo da FFT;
  - *N_AMOSTRAS_DESCARTADAS* = 50 – nº de amostras iniciais a descartar após cada configuração do NCO com uma nova frequência de varrimento;
  - *LB_SINAL* = 50 – largura de banda considerada para o sinal de entrada;
  - *ERRO_INDEX_MAX* = 1 – erro máximo do índice no teste da frequência estimada;
  - *DELTA_FREQ* = 2000 – incremento de frequência no varrimento (*resolution bandwidth*);
  - *FREQ_INICIAL_SWEEPING* = 10694500 – frequência inicial de varrimento.
- Estruturas:
  - *TYPEEspectro* – estrutura para armazenar as características relevantes de cada espectro:
    - *double riscaIndex* – valor pesado para o índice da risca principal em cada espectro;
    - *double potenciaSinal* – potência do sinal em cada espectro;
    - *double potenciaRuido* – potência de ruído estimada em cada espectro.
- Variáveis globais:
  - *float amostrasFFT[N_AMOSTRAS_FFT]* – *array* para armazenar o quadrado do módulo das amostras do espectro calculado por FFT;

▫ *float real_in[N_AMOSTRAS_FFT]* – *array* para armazenar a parte real das amostras recebidas para o cálculo do espectro por FFT;

▫ *float imag_in[N_AMOSTRAS_FFT]* – *array* para armazenar a parte imaginária das amostras recebidas para o cálculo do espectro por FFT;

▫ *float real_out[N_AMOSTRAS_FFT]* – *array* para armazenar a parte real do espectro calculado por FFT;

▫ *float imag_out[N_AMOSTRAS_FFT]* – *array* para armazenar a parte imaginária do espectro calculado por FFT;

▫ *unsigned char espectroIndex* – índice do espectro a ser processado;

▫ *unsigned char espectroPrincipal* – índice do espectro que contém a risca principal entre todos os espectros calculados;

▫ *TYPEEspectro espectro[N_ESPECTROS]* – *array* de estruturas para armazenar as características relevantes dos espectros calculados por FFT.

- Variáveis locais:

  ▫ *float resEspectral* – resolução espectral utilizada no cálculo das FFTs;

  ▫ *unsigned short maxIndex* – índice da risca com maior amplitude em cada espectro;

  ▫ *unsigned short numRiscas* – nº de riscas de sinal a considerar em cada espectro (riscas principal e adjacentes);

  ▫ *short riscaInferior, riscaSuperior* – índices das riscas de limite inferior e superior do espectro a ser processado;

  ▫ *short riscaSinalInferior, riscaSinalSuperior* – índices da primeira e última risca adjacente a considerar para o sinal no espectro a ser processado;

  ▫ *float maxSNR* – índices das riscas de limite inferior e superior do espectro a ser processado;

  ▫ *float limiarSNR* – valor mínimo da relação entre as potências do sinal e do ruído para decidir se se efectua o teste da frequência estimada ou se se repete o varrimento;

  ▫ *float limiarSNR2* – valor mínimo da relação entre as potências do sinal e do ruído para validar a nova frequência estimada durante o teste da frequência estimada anteriormente;

  ▫ *double freqNCO_test* – palavra de configuração correspondente à frequência do NCO durante o teste da frequência estimada.

- Rotinas:

  ▫ *void FFT_sweeping(void)* – rotina que implementa a estimação da frequência do sinal de entrada;

  ▫ *void fft_float (unsigned NumSamples, int InverseTransform, float *RealIn, float *ImagIn, float *RealOut, float *ImagOut)* – rotina que efectua os cálculos da FFT.

Este módulo foi implementado de acordo com o processo descrito anteriormente na secção 3.1 e que está representado mais detalhadamente no fluxograma de *software* da Figura 4.3.

Depois das configurações iniciais do sistema estarem concluídas, o primeiro passo é efectuar a estimação da frequência do sinal de entrada. Assim, tendo a *flag* do estado do sistema o valor *FFT_SWEEPING*, a rotina *RSI_RINT1()* dá início à execução deste módulo, recebendo todas as amostras necessárias para estimar cada um dos espectros. O controlo da recepção das amostras é feito mediante o contador de amostras *amostrasIndex*, que é inicializado a cada grupo de amostras necessárias para o cálculo da FFT (*N_AMOSTRAS_FFT*).

Após, a recepção de cada um destes grupos, o porto McBSP1 é desactivado e é invocada a rotina *FFT_sweeping()* que efectua todos os passos necessários para a estimação e análise do espectro correspondente, incluindo a invocação da rotina *fft_float()* que calcula a FFT, a discriminação da risca principal e ainda a estimação da potência de sinal, da potência de ruído e do valor pesado para o índice desta risca. No final deste processamento, a frequência do NCO é incrementada (*DELTA_FREQ*) e o porto McBSP1 é reactivado para a recepção do grupo de amostras para o cálculo do espectro seguinte.

No entanto deve-se referir que, para para evitar o transiente inicial à saída do detector de fase de cada vez que o porto McBSP1 é reactivado, são descartadas as amostras iniciais num total dado por *N_AMOSTRAS_DESCARTADAS*.

O controlo do número de grupos de amostras recebidos, ou seja, do número de espectros estimados fica a cargo do contador *espectrosIndex*. O número total de espectros do varrimento é dado por *N_ESPECTROS* e quando o contador atinge este valor dá-se por terminado o varrimento. Cabe então à rotina *FFT_sweeping()* a discriminação do espectro principal entre todos os espectros analisados. Se o valor de SNR estimado para o espectro inicial for superior a um determinado limite mínimo (*limiarSNR*), a frequência do sinal de entrada é estimada, a *flag* de estado do sistema é alterada para *FFT_SWEEPING_TEST* e o sistema procede ao teste da frequência estimada. Caso contrário, as condições iniciais deste varrimento são repostas e repete-se o varrimento.

O processo de análise do espectro neste teste é semelhante ao anterior mas além de não ser efectuada a discriminação do espectro principal, a averiguação da fiabilidade do espectro é mais exigente. Além do limite mínimo para considerar o valor de SNR fiável, o desvio entre a primeira risca e a risca detectada como principal neste novo espectro é também limitado a um valor bastante pequeno (*ERRO_INDEX_MAX*).

Se estas condições se verificarem, é estimada novamente a frequência do sinal de entrada e a *flag* de estado do sistema é alterada para *PLL_DIGITAL_START*. É ainda inicializado o valor da saída do AGC (*AGC_out*) com a amplitude estimada do sinal e é guardado o valor da potência média de ruído, que é uma estimativa da densidade espectral de potência de ruído e que será posteriormente necessário no módulo de estimação de CNR. A frequência do NCO é configurada de acordo com o novo valor estimado, que é também o valor configurado como *offset* de frequência do NCO. De seguida procede-se à inicialização dos filtros da cadeia secundária de filtragem e decimação. Se pelo contrário o valor de SNR não for fiável, analogamente ao que se descreveu anteriormente, são repostas as condições iniciais e repete-se o varrimento.

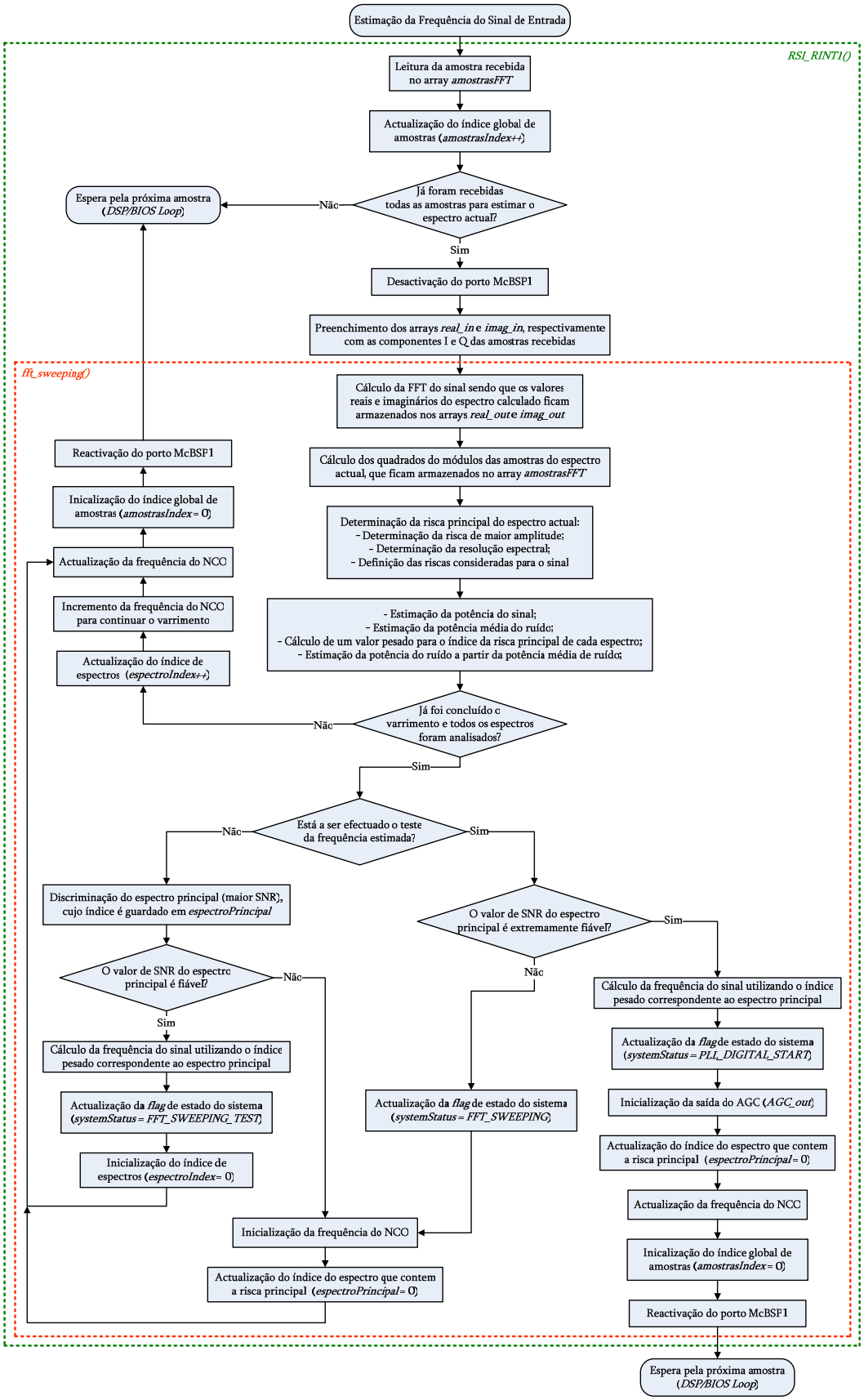

Figura 4.3 – Fluxograma de *software* do módulo de estimação da frequência do sinal de entrada.

4.3. Cadeia Secundária de Filtragem e Decimação

Na implementação do módulo da cadeia secundária de filtragem e decimação foram definidas as seguintes constantes, estruturas, variáveis e rotinas mais relevantes:

- Constantes:
  - *M1* = 40 – factor de decimação do filtro FIR1;
  - *M2* = 5 – factor de decimação do filtro FIR2;
  - *MTOTAL.* = 200 – factor de decimação total dos filtros FIR1 e FIR2;
  - *N_AMOSTRAS_FIR1* = 120 – nº de elementos do *buffer* circular do filtro FIR1;
  - *N_AMOSTRAS_FIR2* = 50 – nº de elementos do *buffer* circular do filtro FIR2;
  - *N_AMOSTRAS_INICIAIS* = 1880 – nº de amostras necessárias para o preenchimento inicial dos *buffers* circulares dos filtros FIR.
- Estruturas:
  - *BUFFER* – estrutura de um *buffer* circular que contém as seguintes variáveis:
    - *unsigned char ii, ir* – pontos de inserção/retirada na/da memória do *buffer*.
- Variáveis globais:
  - *unsigned short amostra_n* – índice de amostras para a cadeia secundária de filtragem e decimação;
  - *short amostraI_out* – amostra I na saída da cadeia de filtragem e decimação;
  - *short amostraQ_out* – amostra Q na saída da cadeia de filtragem e decimação;
  - *BUFFER *structFIR1_I, *structFIR1_Q* – ponteiros para estruturas de *buffers* circulares para as amostras I e Q no filtro FIR1;
  - *BUFFER *structFIR2_I, *structFIR2_Q* – ponteiros para estruturas de *buffers* circulares para as amostras I e Q no filtro FIR2;
  - *float bufferFIR1_I[N_AMOSTRAS_FIR1]* – *array* para armazenar as amostras I à entrada do filtro FIR1;
  - *float bufferFIR1_Q[N_AMOSTRAS_FIR1]* – *array* para armazenar as amostras Q à entrada do filtro FIR1;
  - *float bufferFIR2_I[N_AMOSTRAS_FIR2]* – *array* para armazenar as amostras I à entrada do filtro FIR2;
  - *float bufferFIR2_Q[N_AMOSTRAS_FIR2]* – *array* para armazenar as amostras Q à entrada do filtro FIR2.

- Rotinas:

  ▫ *void filtDecimInic(void)* – rotina que inicializa a cadeia secundária de filtragem e decimação;

  ▫ *void filtDecim(void)* – rotina que implementa a cadeia secundária de filtragem e decimação;

  ▫ *void bufferInit(BUFFER *bp)* – rotina que inicializa os pontos de inserção e retirada do *buffer* circular cuja estrutura é apontada por *bp*.

- Variáveis locais:

  ▫ *float auxI, auxQ* – variáveis auxiliares das rotinas *filtDecimInic()* e *filtDecim()* para os cálculos dos filtros FIR relativos às componentes I e Q.

Antes do sistema iniciar a execução da PLL, é necessário inicializar os filtros FIR da cadeia secundária de filtragem e decimação, uma vez que na situação inicial os *buffers* circulares estão vazios. Uma das soluções possíveis para esta inicialização passaria pelo preenchimento inteligente dos *buffers*, mas tal não seria necessário, uma vez que este preenchimento inicial poderia ser efectuado com amostras reais enviadas pelo AD6620 através da porta série e com a malha aberta. Seriam apenas recebidas as amostras em número suficiente para poder encher os *buffers* circulares do filtro FIR1, num número de vezes necessário para efectuar os cálculos associados a este filtro e preencher os *buffers* circulares do filtro FIR2, com excepção das últimas *M2* posições de memória. Não é necessário preencher todas as posições destes *buffers*, na medida em que a inicialização da cadeia de filtragem e decimação só engloba os cálculos associados ao filtro FIR1. A última posição dos *buffers* do filtro FIR2 só é preenchida depois de recebidas *MTOTAL* amostras após o início da execução da PLL. Só a partir desse instante é que os cálculos associados ao filtro FIR2 começam a ser efectuados, surgindo as primeiras amostras à saída da cadeia de filtragem e decimação.

O processo descrito anteriormente foi implementado de acordo com o fluxograma de *software* que está representado na Figura 4.4. Como se pode observar nesta figura, o controlo do fluxo é incumbido ao índice contador de amostras *amostrasIndex*, sendo que a inicialização dos filtros termina quando este índice atingir o valor dado pela constante *N_AMOSTRAS_INICIAIS*. Nesta última iteração, antes de arrancar com a execução da PLL digital, o índice *amostrasIndex* é inicializado a zero e o valor da *flag* de estado do sistema é alterado de *PLL_DIGITAL_START* para *TESTING_LOCK*. Estes processos, bem como a recepção das amostras e a sua divisão em componentes I e Q, ficam a cargo da rotina *RSI_RINT1()*.

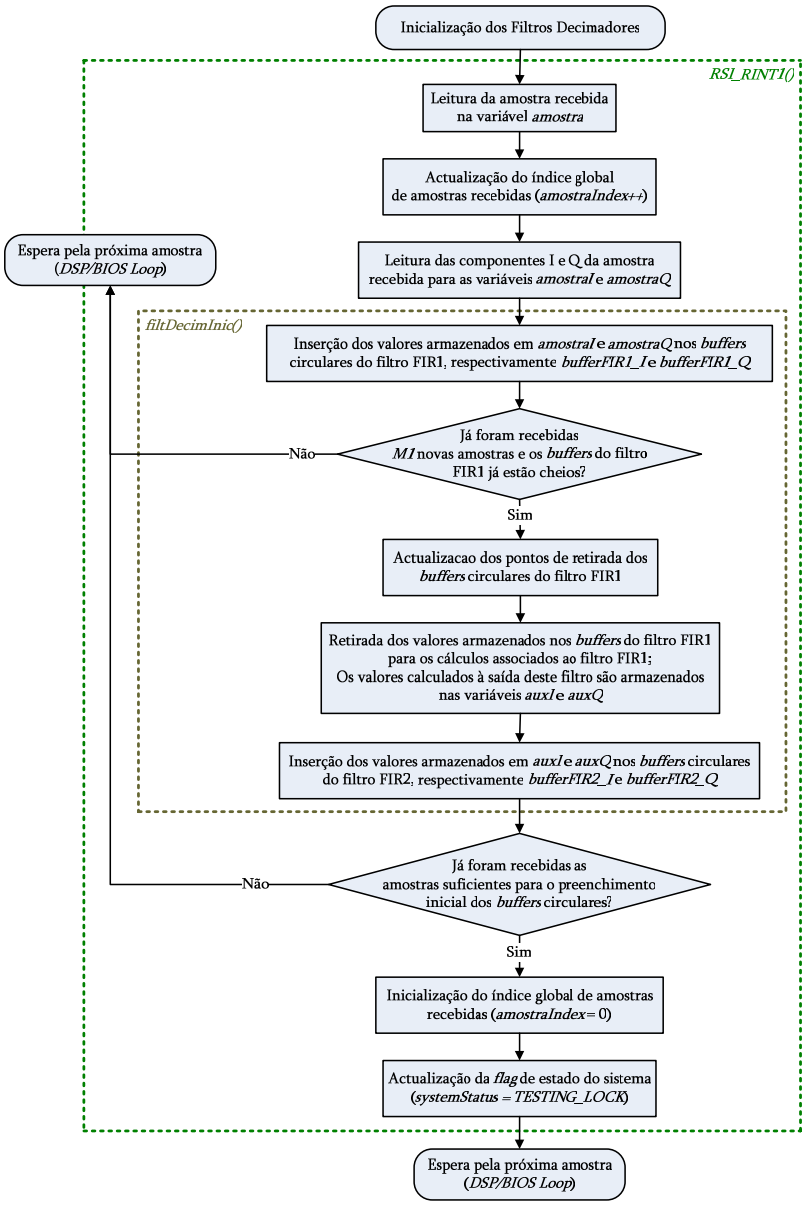

Figura 4.4 – Fluxograma de *software* da inicialização dos filtros da cadeia de filtragem e decimação.

Depois dos *buffers* estarem inicializados, começam a ser calculadas as primeiras amostras à saída do filtro FIR2 de acordo com o fluxograma da Figura 4.5, dando-se início à execução da PLL digital. De salientar o facto de que o processamento descrito neste fluxograma é efectuado continuamente após a inicialização dos *buffers* do filtro FIR. No entanto, nas primeiras vezes que este processamento é efectuado o estado do sistema corresponde ao teste de sincronismo. Só depois deste teste ser concluído com sucesso é que o processamento relativo aos filtros FIR passa a ser feito durante a execução da PLL digital ou durante o "congelamento" do NCO.

Comparando este fluxograma com o anterior, é possível observar que a grande diferença reside no facto do primeiro já englobar os cálculos relativos ao filtro FIR2, bem como a actualização do valor da *flag* de estado do sistema para *PLL_DIGITAL*. O

controlo do fluxo passa agora a estar também a cargo do índice de amostras para a filtragem e decimação *amostra_n*, que define o factor de decimação, ou seja, os instantes em que são efectuados os cálculos dos filtros. De salientar ainda que a inicialização desta variável é efectuada durante a execução do AGC digital (ver secção 4.4).

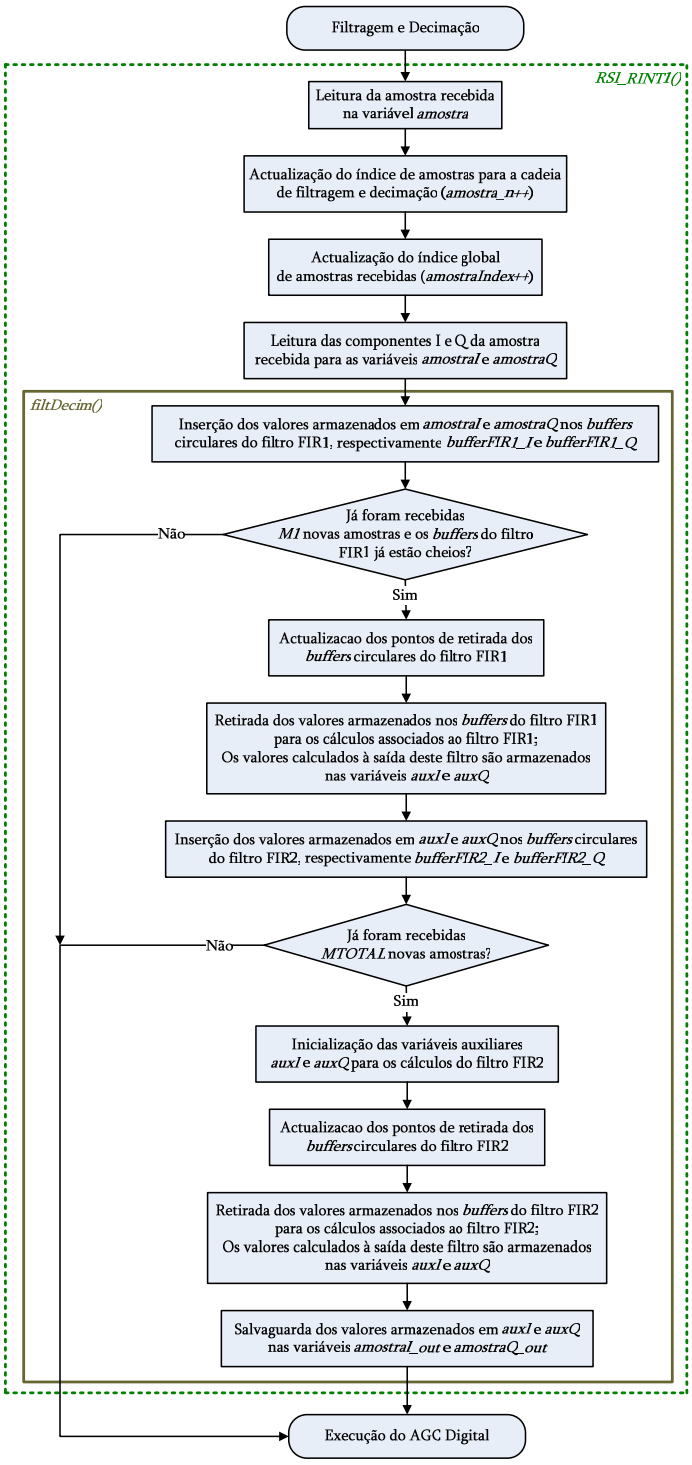

Figura 4.5 – Fluxograma de *software* da implementação dos filtros da cadeia de filtragem e decimação.

4.4. Controlo Automático de Ganho (AGC)

Na implementação do módulo do controlo automático de ganho foram definidas as seguintes variáveis globais e rotinas mais importantes:

- Constantes:

  ▫ *AGC_MAX_OUT* – valor do limite para a saída do AGC, ou seja, para a amplitude detectada.

- Variáveis globais:

  ▫ *short amostraI_AGC* – amostra I na saída do AGC;

  ▫ *short amostraQ_AGC* – amostra Q na saída do AGC;

  ▫ *double AGC_out* – valor estimado pelo AGC para a amplitude do sinal.

- Rotinas:

  ▫ *void AGC_digital(void)* – rotina que implementa o controlo automático de ganho.

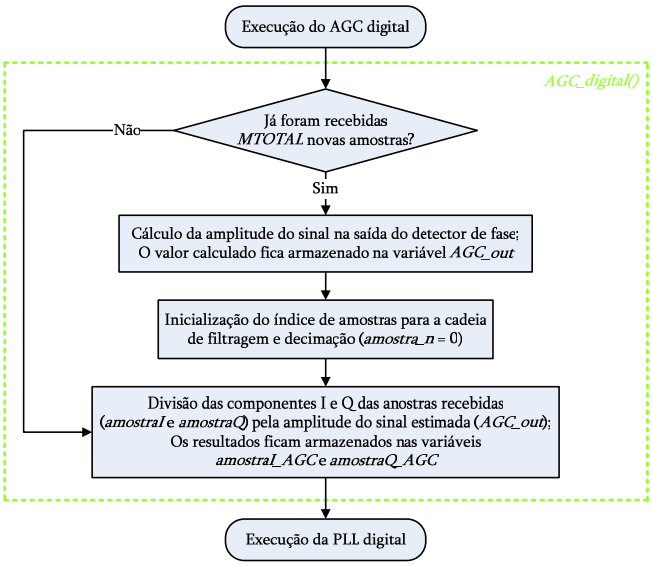

Figura 4.6 – Fluxograma de *software* do módulo de controlo automático de ganho.

Tal como se pode observar no fluxograma de *software* representado na Figura 4.6, a execução do módulo de controlo automático de ganho fica a cargo da rotina *AGC_digital()*, que foi implementada de acordo com o diagrama de blocos da Figura 3.11.

Todavia, é importante salientar que apesar de a amplitude do sinal ser estimada apenas a cada *MTOTAL* amostras recebidas, esta rotina é executada sempre que é recebida uma amostra pela porta série. A frequência com que o AGC estima a amplitude

do sinal de entrada depende então dos filtros usados na cadeia secundária de filtragem e decimação. De acordo com o que se referiu na secção 3.3.1, é esta a frequência que define a largura de banda da AGC.

Importa também referir que nas primeiras iterações, enquanto não foram recebidas *MTOTAL* amostras, o AGC ainda não pode calcular o valor da amplitude do sinal. Desta forma, a variável *AGC_out* terá que ter um valor inicial suficientemente próximo da amplitude do sinal. A solução encontrada para este problema passou por inicializar esta variável durante a estimação da frequência do sinal de entrada, mais propriamente durante o teste da frequência estimada, como se pode ver no fluxograma da Figura 4.3. O valor estimado neste instante seria suficientemente fiável para garantir que o processo do AGC se inicie e decorra sem nenhum problema.

4.5. Indicação de Sincronismo

De seguida apresentam-se as constantes, variáveis globais e rotinas que foram definidas na implementação do módulo de indicação de sincronismo:

- Constantes:

  ▫ *UNLOCKED* = 0 – estado da *flag* de indicação de sincronismo (*lockFlag*) – *loop* fora de sincronismo;

  ▫ *LOCKED* = 1 – estado da *flag* de indicação de sincronismo (*lockFlag*) – *loop* em sincronismo;

  ▫ *MIN_LOCK* = 0.9 – limite mínimo da *running average* da componente I actuada pelo AGC, para considerar a malha em sincronismo.

- Variáveis globais:

  ▫ *unsigned char lockFlag* – *flag* de indicação de sincronismo;

  ▫ *unsigned short indexLOCK* – índice de amostras para a indicação de sincronismo;

  ▫ *double averageLOCK* – valor da *running average* da componente I actuada pelo AGC.

- Rotinas:

  ▫ *void lockIndication(void)* – rotina que implementa a indicação de sincronismo.

Na Figura 4.7 está representado o fluxograma de *software* do módulo de indicação de sincronismo.

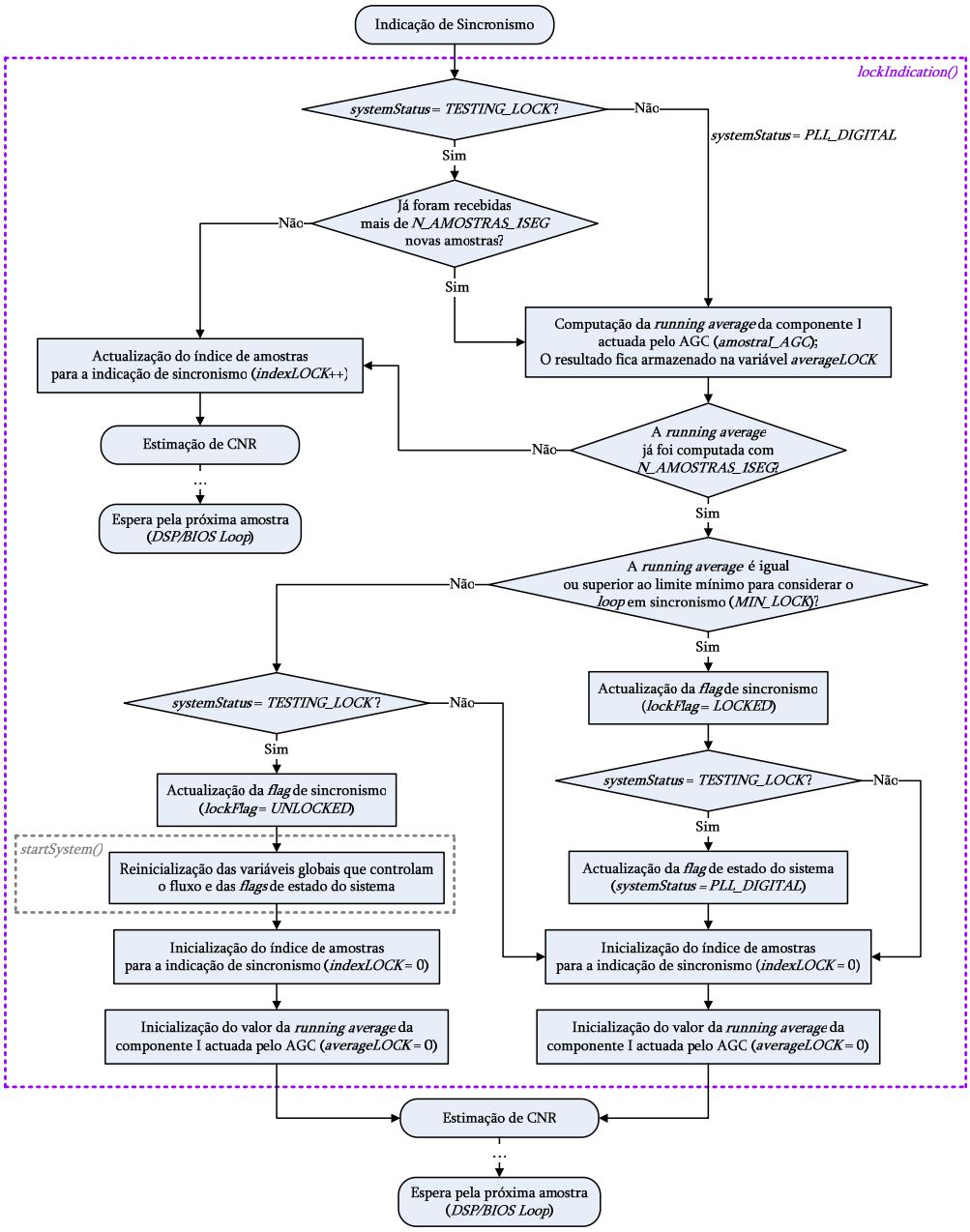

Figura 4.7 – Fluxograma de *software* do módulo de indicação de sincronismo.

Se o sistema estiver a proceder ao teste de sincronismo (*systemStatus* = *TESTING_LOCK*), antes de obter a indicação de sincronismo, a PLL digital é executada durante um período de cerca de 1s (tempo correspondente à recepção de *N_AMOSTRAS_1SEG*), para permitir a aquisição do sincronismo. Após este período de espera, começa a ser computada uma *running average* sobre a componente I actuada pelo AGC, sendo que a PLL continua a ser executada. Se por outro lado o sistema estiver a proceder à execução normal da PLL digital (*systemStatus* = *PLL_DIGITAL*), a *running average* começa a ser executada imediatamente.

Esta *running average* é calculada durante 1s e posteriormente comparada com o valor mínimo para considerar a malha em sincronismo (*MIN_LOCK*). Caso for superior

ou igual a este limite, a *flag* de sincronismo *lockFlag* toma o valor *LOCKED* e, se o sistema estiver a proceder ao teste de sincronismo (no início da execução da PLL ou após a perda de sincronismo), a *flag* do estado de sistema é alterada para *PLL_DIGITAL*. Caso contrário, a *flag* de sincronismo *lockFlag* toma o valor *UNLOCKED* e, se o sistema estiver também a proceder ao teste de sincronismo, é evocada a rotina s*tartSystem()* para repor as condições iniciais para reinicializar o sistema. Antes de terminar, são inicializados os valores do índice para a indicação de sincronismo (*indexLOCK*) e da variável para armazenar a *running average* (*averageLOCK*), na eventualidade de ser necessário executar este teste de sincronismo posteriormente.

4.6. Estimação de CNR e "Congelamento" do NCO

Na implementação do módulo de estimação de CNR e congelamento do NCO foram definidas as seguintes constantes, variáveis globais e rotinas:

- Constantes:

  ▫ *MIN_CNR* = 30 – valor do limite mínimo de CNR para considerar válida a *running average* da palavra de frequência do NCO;

  ▫ *MIN_CNR1* = 27 – valor do limite mínimo de CNR, abaixo do qual o sinal se considera perdido;

  ▫ *MIN_CNR2* = 27 – valor do limite mínimo de CNR para considerar que o sinal recuperou;

  ▫ *MAX_INDEX_FREQ_NCO* = 5 – valor máximo do índice para o cálculo da *running average* da palavra de frequência do NCO;

  ▫ *MAX_FROZEN_NCO_CNT* = 40 – valor do limite para o contador do tempo de espera pela recuperação do sinal.

- Variáveis globais:

  ▫ *unsigned short indexCNR* – índice de amostras para a estimação de CNR;

  ▫ *unsigned char frozenNCOcounter* – contador para o tempo de espera pela recuperação do sinal;

  ▫ *double averageCNR* – valor da *running average* para estimação de CNR;

  ▫ *float CNR* – valor da *running average* de CNR em dB;

  ▫ *double lastFREQ_NCO_WORD* – valor da última palavra de frequência do NCO considerada válida;

  ▫ *double averageFREQ_NCO_WORD* – valor da *running average* da palavra de frequência do NCO;

▫ *unsigned char indexFREQ_NCO_WORD* – índice de amostras para o cálculo da *running average* da palavra de frequência do NCO;

▫ *double densEspecPotRuido* – estimativa da densidade espectral de potência de ruído obtida durante a estimação da frequência do sinal de entrada.

- Variáveis locais:

  ▫ *double auxCNR* – variável local da rotina *CNR_estimation()* com o valor de CNR correspondente à amostra actual.

- Rotinas:

  ▫ *void CNR_estimation(void)* – rotina que implementa a estimação de CNR;

  ▫ *void freeze_NCO(void)* – rotina que implementa o "congelamento" do NCO.

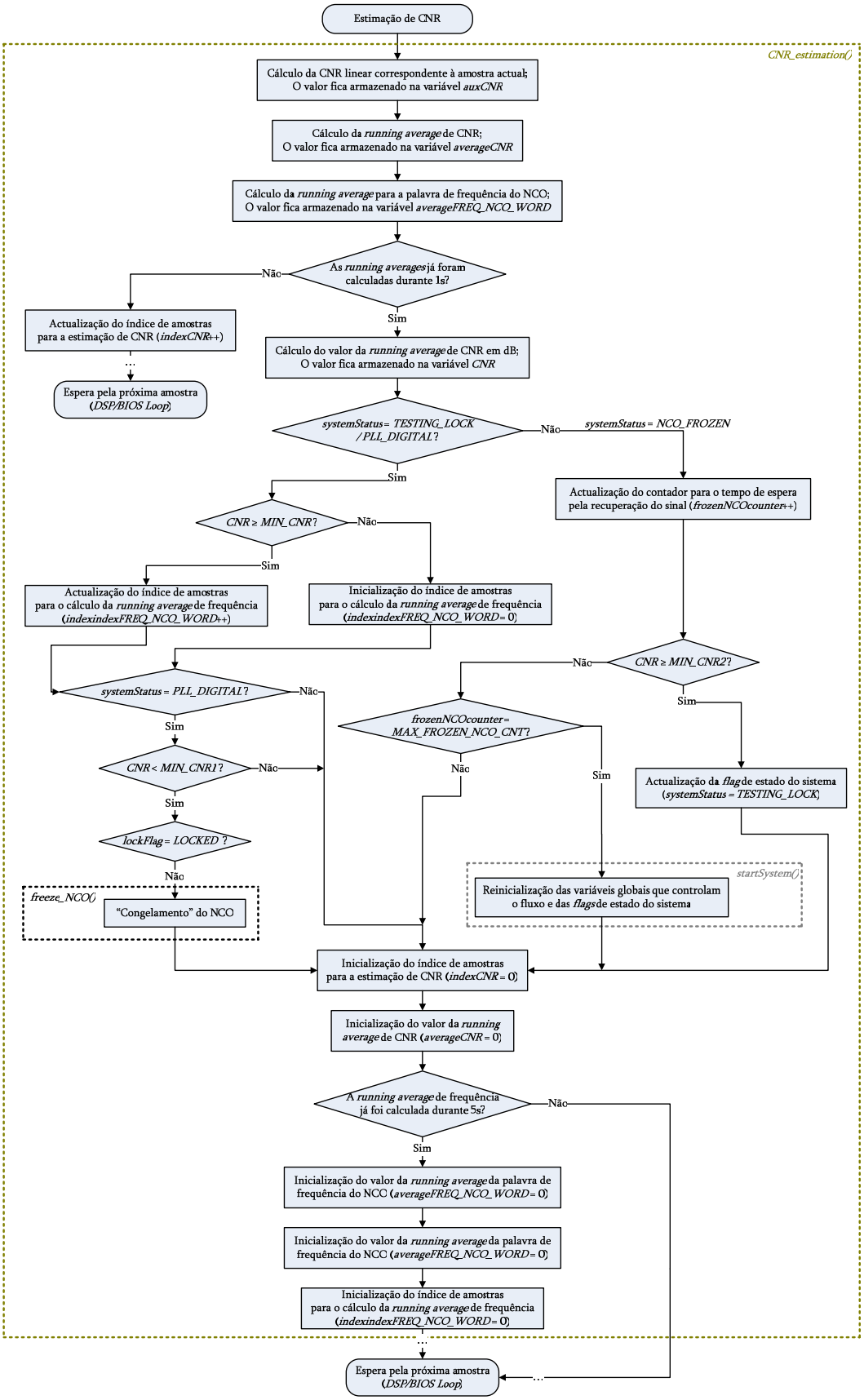

Figura 4.8 – Fluxograma de *software* do módulo de estimação de CNR e "congelamento" do NCO.

Primeiro é calculado o valor linear de CNR correspondente à amostra actual (*auxCNR*), uma vez que uma média calculada sobre valores lineares não corresponde a uma média calculada com valores em dB. Este valor é utilizado posteriormente para calcular a *running average* de CNR (*averageCNR*), que é calculada a cada segundo. De seguida é calculada a *running average* da palavra de configuração de frequência do NCO (*averageFREQ_NCO_WORD*), que será utilizada para o "congelamento do NCO" quando for necessário. Esta *running average* é calculada a cada 5 segundos, mas necessita de ser validada a cada segundo, de acordo com o valor de CNR estimado a cada segundo.

Neste módulo, o controlo do fluxo fica a cargo dos índices de amostras *indexCNR* e *indexFREQ_NCO_WORD*, em que o primeiro controla o número de amostras com que é efectuada a *running average* de CNR (*N_AMOSTRAS_1SEG*), enquanto que o segundo controla o número de vezes em que são recebidas *N_AMOSTRAS_1SEG* para o cálculo da *running average* da *word* de frequência (*MAX_INDEX_FREQ_NCO*).

Durante o cálculo desta última *running average*, se o valor de CNR estimado a cada segundo for abaixo de um determinado limite para a considerar válida (*MIN_CNR*), as variáveis *averageFREQ_NCO_WORD* e *indexFREQ_NCO_WORD* são inicializadas a zero e inicia-se um novo cálculo. Este processo repete-se até que até que tenham sido recebidas as amostras correspondentes a um período de 5s. Durante este período, o índice *indexFREQ_NCO_WORD* vai sendo incrementando a cada segundo, desde que se obtenha sempre uma indicação válida de CNR. Neste caso, o valor da *running average* da palavra de frequência é considerado fiável para o "congelamento" do NCO e a variável *lastFREQ_NCO_WORD* é actualizada com este valor, sendo que as variáveis *indexFREQ_NCO_WORD* e *averageFREQ_NCO_WORD* são ainda inicializadas a zero tendo em vista o próximo cálculo. Como é óbvio, a actualização desta variável só ocorre quando a *flag* de estado do sistema contém o valor *PLL_DIGITAL* ou *TESTING_LOCK*.

A escolha do valor de 5s tomou por referência o objectivo de conseguir uma variância de frequência de aproximadamente 1Hz, o que implica qua a *running average* seja calculada durante este período de tempo.

Quando a *flag* de estado do sistema contém o valor *PLL_DIGITAL*, o valor da running average de CNR é comparado a cada segundo com o valor mínimo dado por MIN_CNR1. Quando a *running average* é inferior a este limite, o sistema verifica se existe indicação de sincronismo (através da *flag* de sincronismo). Se não houver indicação positiva de sincronismo, o sistema procede ao "congelamento" do NCO (ver Figura 4.9). De seguida, as variáveis *indexCNR* e *averageCNR* são inicializadas a zero, preparando-as para o próximo cálculo.

Por outro lado, quando a *flag* de estado do sistema contém o valor *TESTING_LOCK*, o sistema limita-se a calcular as *running averages* de CNR e frequência. Analogamente ao que se referiu anteriormente, as variáveis *indexCNR* e *averageCNR* são também inicializadas.

Quando o NCO está "congelado", o sistema espera pela recuperação do sinal, testando apenas o valor *running average* de CNR, ou seja, comparando-o com o valor mínimo para considerar que o sinal recuperou (*MIN_CNR2*). Quando a CNR atinge este valor mínimo, a *flag* de estado do sistema é alterada para o valor *TESTING_LOCK*, pelo que o sistema irá proceder ao teste de sincronismo. O tempo de recuperação do sinal é definido pelo contador *frozenNCOcounter*, que é incrementado a cada segundo e a partir do instante em que o NCO é "congelado". Quando este contador atinge o valor máximo dado por *MAX_FROZEN_NCO_CNT*, é evocada a rotina *startSystem()* para reinicializar o sistema.

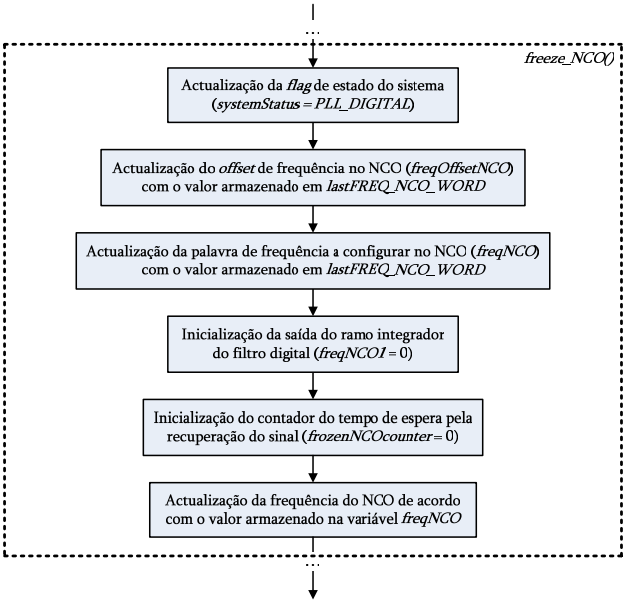

Figura 4.9 – Fluxograma de *software* do "congelamento" do NCO.

Como se pode ver no fluxograma da Figura 4.9, a rotina *freeze_NCO()* é responsável pelo "congelamento" do NCO, começando por actualizar a *flag* de estado do sistema para o valor *NCO_FROZEN*. As variáveis com as palavras de frequência e do *offset* de frequência do NCO, respectivamente *freqNCO* e *freqOffsetNCO*, são actualizadas com a última palavra fiável de frequência contida em *lastFREQ_NCO_WORD*. De seguida a saída do ramo integrador do filtro da PLL é inicializada a zero (*freqNCO1*), assim como o contador do tempo de espera para a recuperação do sinal (*frozenNCOcounter*). Por fim, a frequência do NCO é configurada com o a palavra armazenada em *freqNCO*, sendo esta a frequência à qual o NCO é "congelado".

4.7. PLL

Na implementação do módulo da PLL digital foram definidas as seguintes constantes, variáveis globais e rotinas:

- Constantes:

  ▫ *const double G_AGC* = 18500 – valor fixo pelo qual a componente Q actuada pelo AGC é multiplicada para implementação dos cálculos associados ao filtro da PLL. É a partir deste valor que são determinadas as constantes deste filtro;

  ▫ *const double C1* = 0.002824363694 – valor para a constante C_1 do ramo integrador do filtro digital da PLL;

  ▫ *const double C2* = 0.206820926922 – valor para a constante C_2 do ramo proporcional do filtro digital da PLL.

- Variáveis globais:

  ▫ *double freqNCO1* – valor da saída do ramo integrador do filtro digital da PLL;

  ▫ *double freqOffsetNCO* – valor de *offset* de frequência do NCO que corresponde à frequência do NCO no início da execução da PLL.

- Rotinas:

  ▫ *void PLL_digital(void)* – rotina que implementa a PLL digital.

Na Figura 4.10 está representado o fluxograma de *software* do módulo da PLL digital.

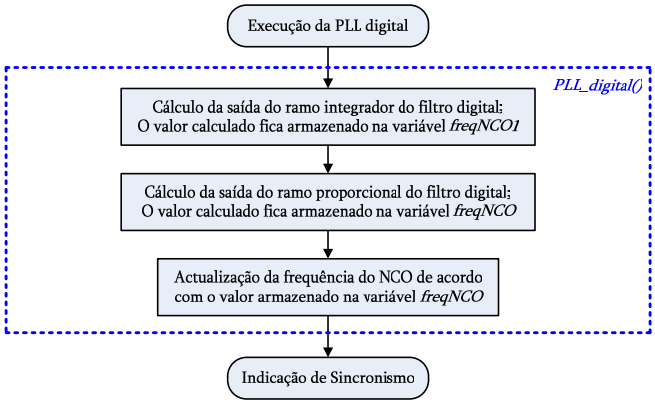

Figura 4.10 – Fluxograma de *software* do módulo da PLL digital.

Como se pode observar no fluxograma anterior, o primeiro passo consiste na aplicação do filtro digital sobre a componente Q actuada pelo AGC e multiplicada pelo ganho *G_AGC*.

Primeiro é calculado o valor à saída do ramo integrador do filtro digital:

$$freqNCO1_n = freqNCO1_{n-1} + amostraQ\_AGC_n \times G\_AGC \times C1 . \quad (4.1)$$

De seguida, o valor à saída do ramo integrador é somado ao valor à saída do ramo proporcional e ao *offset* de frequência do NCO, sendo que o valor à saída do filtro digital é dado por:

$$freqNCO_n = freqNCO1_n + amostraQ\_AGC_n \times G\_AGC \times C2 + freqOffsetNCO\,. \qquad \textbf{(4.2)}$$

O último passo consiste em actualizar a frequência do NCO de acordo com valor armazenado na variável *freqNCO*. Esta variável contém a palavra para actualizar a frequência do NCO na iteração *n*.

Como se referiu na secção 4.2, antes de de iniciar a execução da PLL, o valor inicial da frequência e do *offset* da frequência do NCO corresponde ao valor estimado no módulo de estimação de frequência do sinal de entrada. Este valor inicial é calculado durante o arranque do sistema ou em situações de perda de sincronismo. Importa ainda referir que o saída do ramo integrador é inicializada a zero.

4.8. FLL

O módulo da FLL digital foi implementado numa versão diferente do programa final e por isso não é mencionado nas secções anteriores deste capítulo. No entanto, o enquadramento deste módulo na estrutura do programa seria exactamente igual ao do módulo da PLL digital, com a excepção do módulo de indicação de sincronismo. A abordagem utilizada neste último módulo seria totalmente ineficiente com a FLL. Neste caso, efectuar uma média sobre a componente I actuada pelo AGC seria totalmente inútil, tendo em vista a obtenção de uma indicação fiável de sincronismo. Esta será então uma questão a discutir futuramente.

Na implementação deste módulo foram definidas as seguintes constantes, variáveis globais e rotinas:

- Constantes:

  ▫ *const double G_AGC* = 18500– valor fixo pelo qual a componente Q actuada pelo AGC é multiplicada para implementação dos cálculos associados ao filtro da FLL. É a partir deste valor que são determinadas as constantes deste filtro;

  ▫ *const double C1* = 0.00000066659848/0.00000133501541 – valor para a constante C_1 do filtro digital da FLL com B_L = 5/10Hz;

  ▫ *const double C2* = 0.00000066568926/0.00000133137852 – valor para a constante C_2 do filtro digital da FLL com B_L = 5/10Hz;

- Variáveis globais:

  ▫ *double freqNCO1* – valor à saída do filtro digital da FLL na iteração actual (a menos do *offset*);

  ▫ *double f_nco1*, *f_nco2* – valores à saída do filtro digital (a menos do *offset*) nas duas iterações anteriores à iteração actual;

  ▫ *double out_FD*, *out_FD1* – valores à saída do detector de frequência na iteração actual e na iteração anterior;

  ▫ *double amostraI_prev* – a amostra da componente I actuada pelo AGC na iteração anterior à actual;

  ▫ *double amostraQ_prev* – a amostra da componente Q actuada pelo AGC na iteração anterior à actual;

  ▫ *double freqOffsetNCO* – valor de *offset* de frequência do NCO que corresponde à frequência do NCO no início da execução da FLL.

- Rotinas:

  ▫ *void FLL_digital(void)* – rotina que implementa a FLL digital.

Na Figura 4.11 está representado o fluxograma de *software* do módulo da PLL digital.

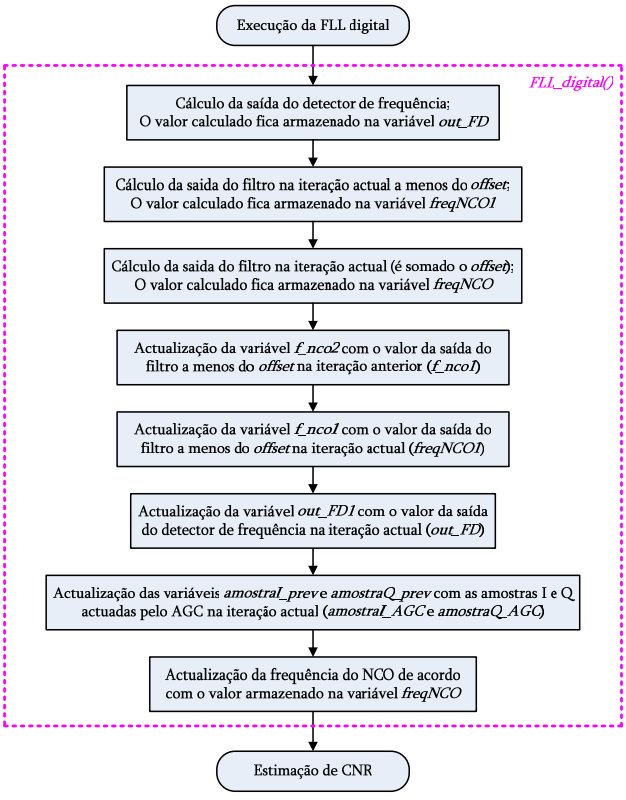

Figura 4.11 – Fluxograma de *software* do módulo da FLL digital.

Como se pode observar no fluxograma anterior, o primeiro passo consiste no cálculo do valor à saída do detector de frequência, utilizando as amostras actuais e da iteração prévia das componentes I e Q actuadas pelo AGC e multiplicadas pelo ganho *G_AGC*:

$$\begin{aligned} out\_FD_n = {} & amostraI\_prev_n \times G\_AGC \times amostraQ\_AGC_n \times G\_AGC \\ & - amostraQ\_prev_n \times G\_AGC \times amostraI\_AGC_n \end{aligned} . \quad \textbf{(4.3)}$$

De seguida, o filtro digital da FLL é aplicado sobre as saídas do detector de frequência. *G_AGC*, de acordo com a seguinte equação:

$$freqNCO1_n = 2 \times f\_nco1_n - f\_nco2_n + out\_FD_n \times C1 - out\_FD1_n \times C2 , \quad \textbf{(4.4)}$$

onde:

$$out\_FD1_n = out\_FD_{n-1} . \quad \textbf{(4.5)}$$

$$f\_nco2_n = f\_nco1_{n-1} = freqNCO1_{n-2} , \quad \textbf{(4.6)}$$

$$f\_nco1_n = freqNCO1_{n-1} . \quad \textbf{(4.7)}$$

De seguida, o valor calculado é somado ao *offset* de frequência do NCO, sendo que o valor à saída do filtro digital é dado por:

$$freqNCO_n = freqNCO1_n + freqOffsetNCO . \quad \textbf{(4.8)}$$

O último passo consiste em actualizar a frequência do NCO de acordo com valor armazenado na variável *freqNCO*. Esta variável contém a palavra para actualizar a frequência do NCO na iteração *n*.

Analogamente à PLL digital, antes de de iniciar a execução da FLL, o valor inicial da frequência e do *offset* da frequência do NCO corresponde ao valor estimado no módulo de estimação de frequência do sinal de entrada. Este valor inicial é calculado durante o arranque do sistema ou em situações de perda de sincronismo. Importa ainda referir que o saída do ramo integrador é inicializada a zero.

4.9. Optimização de *Software*

São inúmeras as potencialidades disponibilizadas pela DSP para optimização de *software*, com o objectivo de melhorar a velocidade de execução e/ou reduzir o tamanho do código. O compilador de C/C++ pode desempenhar diversas tarefas de optimização, como por exemplo a simplificação de ciclos, a execução de ciclos em paralelo ou *software pipelining*, a reordenação de instruções ou a alocação de variáveis para registos.

Existem três fases no desenvolvimento do código, de acordo com o fluxograma e com a lista de tarefas presente em [9, cap. 1, págs. 4-6]. A primeira fase corresponde à escrita do código sem grandes preocupações com o funcionamento da DSP. Caso o código não seja eficiente, procede-se para a segunda fase para refinar e optimizar o código em C, adicionando intrínsecas, directivas e outras técnicas descritas em [10]. Se ainda assim o código não for eficiente, procede-se então à escrita dos excertos de código mais exigentes em *linear assembly* e respectiva optimização na terceira e última fase.

Existem dois tipos de optimização nomeadamente as de baixo nível, que são optimizações mais específicas efectuadas no próprio código, e as de alto nível que são efectuadas pelo optimizador do compilador. A maneira mais simples para invocar optimizações de alto nível para obter um desempenho óptimo, passa por recorrer à opção *–o<n>* do programa *cl6x* do compilador, em que *n* representa o nível ou grau de optimização desejado (entre 0 e 3). Existem assim quatro níveis de optimização, sendo que a optimização é melhor do nível 0 para o nível 3.

Em [10, pág. 54] é possível consultar a longa lista de optimizações associadas aos vários níveis de optimização. À opção *–o3* estão associadas todas estas optimizações de alto nível, de entre as quais a mais importante para este trabalho é o *software pipelining*.

O *software pipelining* é um tipo de optimização que permite agendar as instruções incluídas num ciclo de modo a que múltiplas iterações sejam executadas em simultâneo. Quando o compilador executa este tipo de optimização, a informação relativa ao *software-pipelined loop* é armazenada sobre a forma de comentários no ficheiro "*.asm*" correspondente. O tipo de informação armazenada é descrita em [10, págs. 57-58] e serve para analisar exaustivamente a forma como o ciclo é *software-pipelined*. No entanto, esta informação só existe se o ciclo for qualificado para *software pipelining*. Caso isto não se verifique, em vez deste tipo de informações, são imprimidas mensagens que identificam a causa da desqualificação do ciclo.

De salientar ainda que, de acordo com [9, cap. 2, pág. 2], devem-se ter alguns cuidados na utilização dos vários tipos de variáveis. Quanto às variáveis inteiras deve-se sempre que possível utilizar variáveis do tipo *short* (16 bits) em detrimento do tipo *int* (32 bits), já que estas permitem uma utilização mais eficiente do multiplicador de 16 bits na série C6000 (apenas um ciclo para variáveis *short* contra 5 ciclos para variáveis *int*). Uma outra consideração tem a ver com a execução de instruções em vírgula flutuante em dispositivos da série C6700. Nestes casos deve-se sempre seleccionar a opção *–mv6700* no compilador, já que assim as instruções irão ser executadas mais eficientemente no *hardware* específico para vírgula flutuante, em vez de serem executadas com o *hardware* para vírgula fixa. Isto é de facto uma exigência já que, como se referiu na 3.2, os cálculos dos filtros da cadeia secundária de filtragem e decimação vão ser executados em vírgula flutuante.

A melhor optimização possível é assim uma tarefa morosa que necessita de conhecimentos mais especializados sobre a arquitectura da DSP. Como se poderá ver mais adiante na secção 5.2, a optimização oferecida pelo compilador levou, com uma enorme margem de folga, à execução do programa em tempo real e com os recursos disponíveis pelo que não se aprofundou o assunto.

5. Depuração de *Software*

Neste capítulo, estão descritos os testes mais importantes para a depuração de *software*, pela ordem com que foram executados. Assim, na secção 5.1 descrevem-se os primeiros testes que foram efectuados sobre o código relativo à estimação da frequência do sinal de entrada, enquanto que na secção 5.2 estão descritos os testes para o código de implementação dos filtros da cadeia secundária de filtragem e decimação. Posteriormente, na secção 5.3 descreve-se a depuração de *software* efectuada para o módulo do controlo automático de ganho. O passo seguinte consistiu em efectuar alguns testes para a depuração do módulo de indicação de sincronismo e para o módulo de estimação de CNR e congelamento do NCO, que estão descritos respectivamente nas secções 5.4 e 5.5.

Depois de terminado o *debugging* de *software*, o sistema passou a estar completamente funcional e pronto para a realização das tarefas implicadas neste projecto.

5.1. Estimação da Frequência do Sinal de Entrada

O primeiro passo para a depuração de *software* relativa ao módulo da estimação da frequência do sinal de entrada, foi testar a função disponibilizada pela biblioteca da DSP (*dsp67x.lib*) para o cálculo da FFT – *DSPF_sp_cfftr2_dit*. No entanto, a única versão disponível desta biblioteca foi construída numa versão mais recente do *Code Composer Studio* do que a utilizada neste trabalho. Após uma tentativa falhada de reconstruir a biblioteca nesta última versão, foi necessário recorrer ao código equivalente da função em linguagem C, disponibilizado através do manual da biblioteca [manual da dsplib, pág. 47, 48].

Os testes efectuados consistiram em gerar o sinal de entrada complexo, nomeadamente gerando sinusóides de frequências conhecidas para a parte real e para a parte imaginária, através das funções *sin* e *cos*. De seguida efectuava-se o cálculo da FFT sobre o sinal de entrada gerado, utilizando a função *DSPF_sp_cfftr2_dit*, tal como vem referido em [11, cap. 4, págs. 13-15].

No entanto, ao contrário do que se poderia pensar, apesar dos resultados obtidos estarem muitas vezes de acordo com o que se esperava, nem sempre isto acontecia. Mais tarde, após uma pesquisa na *internet*, descobriu-se que o código equivalente da função em linguagem C apresentava erros, o que explicaria a estranheza dos resultados obtidos.

Assim, recorreu-se à função *fft_float*, cujo código em C foi retirado de [12], para o cálculo da FFT, mas primeiro foi necessário repetir os testes efectuados para a função *DSPF_sp_cfftr2_dit*.

Depois de efectuados os testes, os primeiros resultados observados estavam correctos mas invertidos em relação aos resultados obtidos no *Matlab*. Posteriormente foi possível verificar que ao trocar a posição dos *arrays real_out* e *imag_out* na chamada da rotina *fft_float*, os resultados eram observados correctamente.

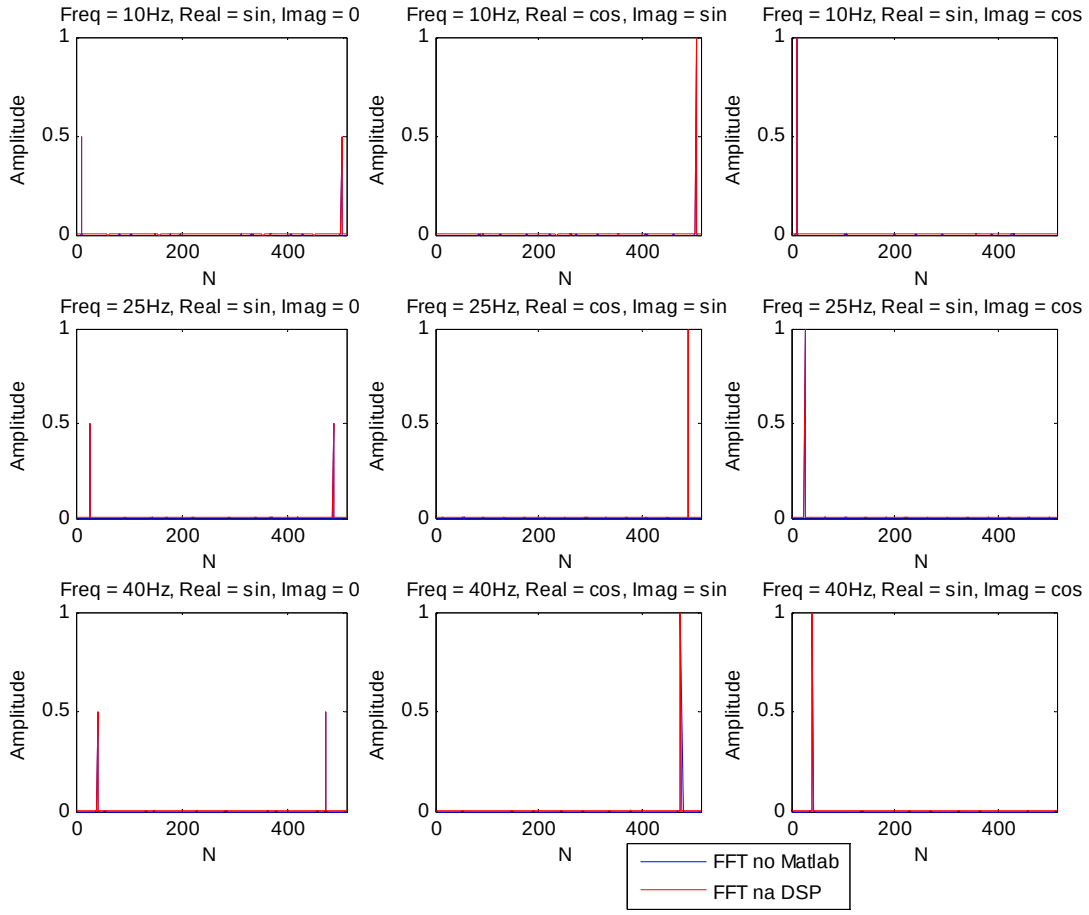

Figura 5.1 – Gráficos com os espectros obtidos nos primeiros testes da função que calcula a FFT.

Na Figura 5.1 estão representados os espectros obtidos após a execução dos testes, juntamente com os espectros obtidos com a função *fft* do *Matlab*. Estes resultados estão de acordo com o que se esperava já que os espectros observados se sobrepõem quase na perfeição.

Figura 5.2 – Gerador de frequências da *Marconi Instruments* – modelo 2022.

Depois dos testes anteriores terem sido efectuados com sucesso, o passo seguinte foi testar novamente a função mas agora recorrendo ao gerador de frequências apresentado na Figura 5.2, para gerar o sinal de entrada. Foram efectuados testes com várias frequências de entrada, que consistiram em arrancar o sistema com a malha aberta e posteriormente recolher 512 amostras à saída do detector de fase para o cálculo da FFT.

Os resultados destes testes estão representados na Figura 5.3, onde se pode observar novamente a total sobreposição entre os espectros obtidos na DSP e no *Matlab*, o que permitiu confirmar a validade da função utilizada para a estimação espectral. Para observar os resultados recorreu-se ao *m-file* que pode ser consultado em Anexos A.

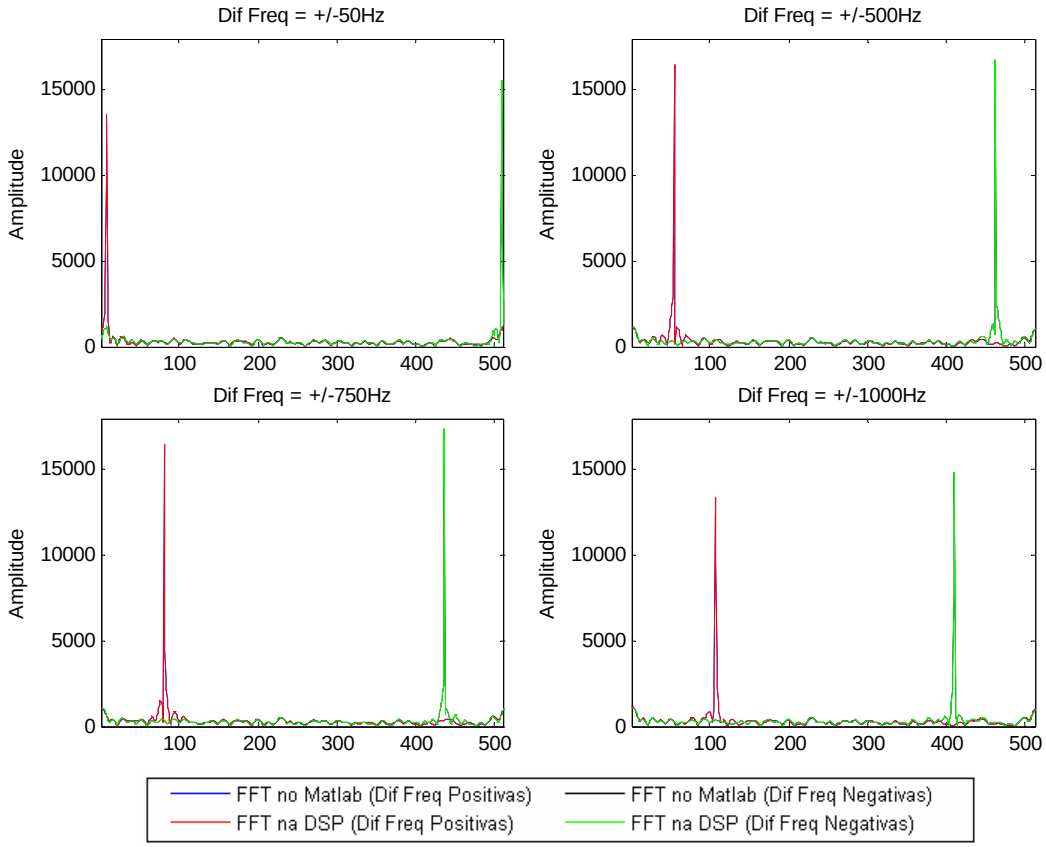

Figura 5.3 – Gráficos com os espectros obtidos nos segundos testes da função que calcula a FFT.

Posteriormente testou-se o código para o módulo de estimação da frequência do sinal de entrada, ao qual se adicionaram simples linhas de código para imprimir os valores das variáveis mais relevantes durante a execução. Assim, com um sinal de entrada com uma frequência de 10.7001MHz, executou-se o código com os seguintes parâmetros:

- Frequência de varrimento inicial: 10.6945MHz;
- Incremento de frequência: 2000Hz;
- Número de espectros: 10;
- Número de amostras para análise espectral: 256.

No entanto, este primeiro teste não teve qualquer êxito, já que além da frequência estimada no final não estar correcta, o espectro principal era também discriminado incorrectamente. Aumentou-se a resolução espectral para 512 amostras e repetiu-se o teste mas os resultados continuaram incorrectos. Isto devia-se ao facto de que nas situações em que o varrimento era feito longe da frequência do sinal de entrada, a potência de ruído estimada tinha um valor muito baixo e consequentemente a SNR toma valores demasiadamente elevados ao contrário do que se esperaria. Estes resultados revelaram então a necessidade de efectuar os testes numa situação mais realista, ou seja, com um sinal de entrada afectado por ruído, o que conduziria a um espectro ao redor do sinal bastante mais *flat*.

Por outro lado, uma maneira de evitar a interferência destes improváveis valores elevados de SNR na discriminação do espectro principal, seria ignorar todos os espectros aos quais correspondessem valores muto reduzidos das estimativas da potência de ruído.

Figura 5.4 – Gerador de sinal da *Hewlett Packard* – modelo 8648B.

Figura 5.5 – Gerador de ruído da *Hewlett Packard* – modelo HO1-3722A.

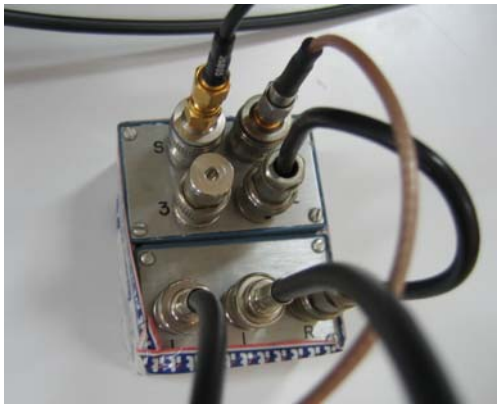

Figura 5.6 – Misturador RF (ZAD3 da *Minicircuits*) e *splitter* de 3 portas (ZSC3-1 da *Minicircuits*).

Para gerar um sinal com CNR = 55dB/Hz, que simulasse o sinal proveniente do satélite, recorreu-se a um outro gerador de sinal (ver Figura 5.4), a um gerador de ruído na banda base (ver Figura 5.5) e a um misturador RF e um *power splitter* (ver Figura 5.6), conforme mostra o esquema de montagem da Figura 5.7. A ideia seria fazer a conversão do ruído para 10.7MHz com o auxílio do misturador e somá-lo posteriormente com o sinal do outro gerador usando o *power splitter*. O sistema funcionou adequadamente excepto com uma pequena contrariedade. Como o ruído DC é convertido com um oscilador cuja frequência é próxima do sinal de teste, o vector de ruído observa-se a "circular" à frequencia diferença destes osciladores. De qualquer forma a maioria dos testes foram efectuados com este arranjo.

Posteriormente foi usada um díodo gerador de ruído de banda larga com vários amplificadores passabanda em cascata contudo, neste caso, a potência de ruído tinha uma largura de banda maior que a pretendida.

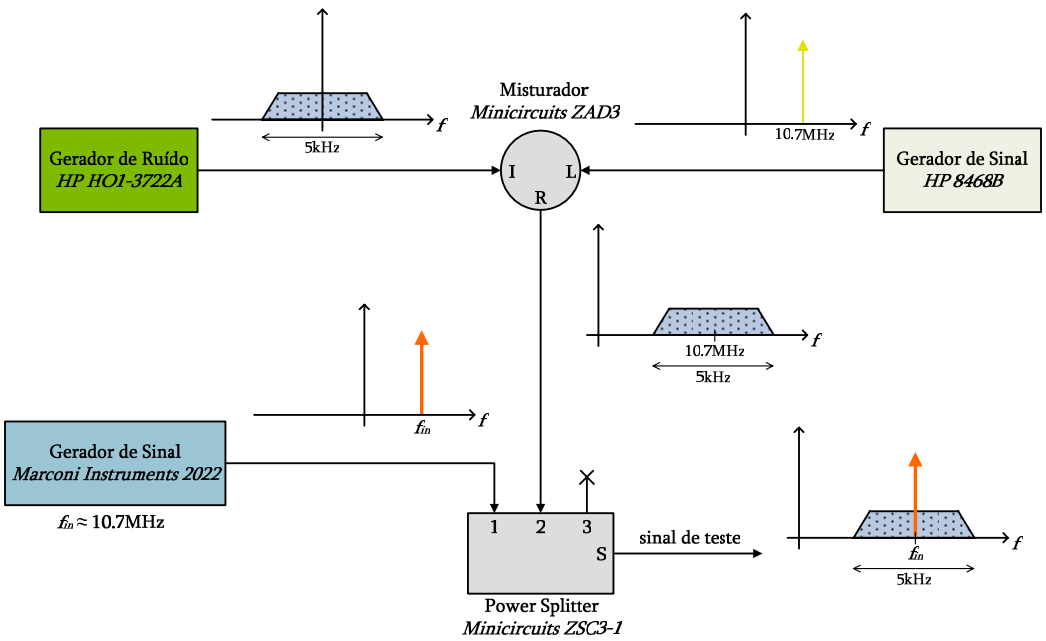

Figura 5.7 – Esquema de montagem utilizado para gerar o sinal com ruído.

Figura 5.8 – Analisador de espectros da *Hewlett Packard* – modelo 8593E.

Para conferir se o sinal com ruído está a ser gerado de acordo com o que se pretende, recorreu-se ao analisador de espectros (ver Figura 5.8). O espectro do ruido gerado está representado na Figura 5.9, enquanto que o espectro do sinal com o ruído adicionado pode ser observado na Figura 5.10. Estes espectros foram observados com as seguintes características no analisador: "*span*" = 25kHz, "RBW" = 100Hz, "VBW" = 100Hz, "*frequency center*" = 10.7MHz e "*video average*" = 15.

Analisando o espectro foi possível estimar a SNR do sinal gerado, medindo a diferença de potência entre o pico do sinal e o patamar de ruído. As potências seriam então ajustadas de forma a obter-se um valor de SNR próximo do valor ao qual correspondem 55dB de CNR. Este ajuste foi feito tendo em conta que o valor da CNR pode ser aproximado por:

$$CNR = SNR|_{RBW} + 10\log_{10}(RBW)\,, \tag{5.1}$$

onde RBW representa a *resolution bandwidth* do analisador de espectros que é ligeiramente menor que a largura de banda de ruído.

De acordo com a equação (5.1), o sinal foi então ajustado para uma $SNR|_{100Hz}$ de 35dB.

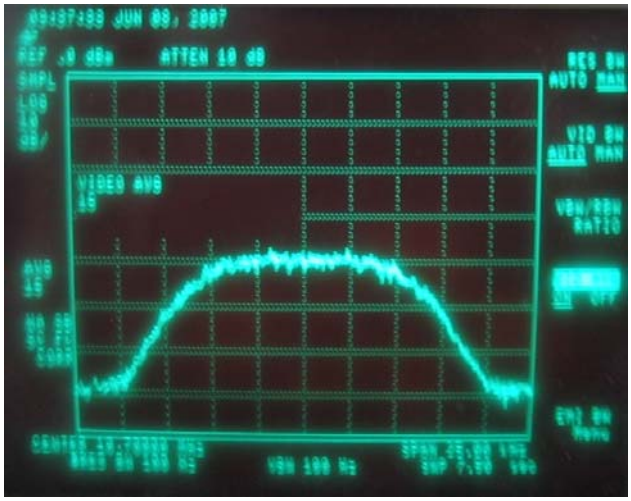

Figura 5.9 – Espectro do ruído gerado.

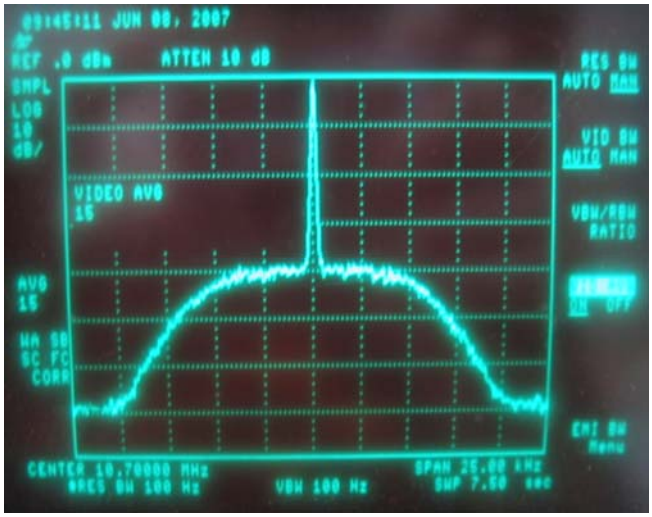

Figura 5.10 – Espectro do sinal com o ruído adicionado.

Averiguou-se ainda que o isolamento entre o oscilador auxiliar e a cadeia de sinal de teste corresponde a aproximadamente 60dB, o que garante uma boa margem mesmo que o sinal de entrada seja atenuado em 30dB.

Voltou-se então a executar o código com os mesmos parâmetros do teste anterior mas com o sinal gerado com ruído. Os resultados obtidos podem ser consultados na Tabela 5.1, onde a resolução espectral, ou seja, o espaçamento entre riscas consecutivas corresponde a aproximadamente 9.54Hz.

Espectro	Frequência NCO (MHz)	Risca Principal	Riscas de Sinal	Índice Pesado	Potência do Sinal ($\times 10^6$)	Potência do Ruído ($\times 10^6$)	SNR (dB)
0	10.694500	77	74-80	76.82	115601.3483	14478.2986	9.02
1	10.696500	379	376-382	379.04	181701.8762	32301.1961	7.51
2	10.698500	169	166-172	169.00	39065749.6105	40361.6990	29.86
3	10.700500	471	468-474	471.20	76829831.0697	1692214.083	26.57
4	10.702500	262	259-265	261.79	438696.2647	34466.1149	11.05
5	10.704500	52	49-55	52.08	222296.4890	23624.6710	9.74
6	10.706500	354	351-357	353.98	210429.6700	7990.4442	14.21
7	10.708500	144	141-147	144.38	62166.3898	1578.4021	15.95
8	10.710500	447	444-450	447.35	1758.2033	636.6802	4.41
9	10.712500	237	234-240	237.01	38672.0594	313.3646	20.91
Teste da Frequência Estimada Durante o Varrimento							
Espectro Principal	**Frequência NCO (MHz)**	**Risca Principal**	**Riscas de Sinal**	**Índice Pesado**	**Potência do Sinal ($\times 10^6$)**	**Potência do Ruído ($\times 10^6$)**	**SNR**
2	10.700112	0	509-511, 0-3	0.01	79414070.8045	66742.7857	30.75
Frequência do Sinal de Entrada (MHz)					**Frequência Estimada (MHz)**		
10.700100					10.700112		

Tabela 5.1 – Resultados dos testes da estimação da frequência do sinal de entrada.

Como se pode ver, apesar da frequência ter sido estimada com um valor muito próximo do valor esperado, o espectro principal não foi escolhido correctamente já que para o espectro nº3 a frequência do NCO era a mais próxima da frequência do sinal de entrada. Esta escolha errada deveu-se ao facto de que no espectro nº2 o sinal aparece no flanco descendente do filtro RCF, ao passo que no espectro nº3 está contido na banda de passagem deste filtro. Esta situação está representada no esquema da Figura 5.11, onde se pode observar que no espectro nº2, tanto o sinal como o ruído são afectados por um ganho inferior a 1.

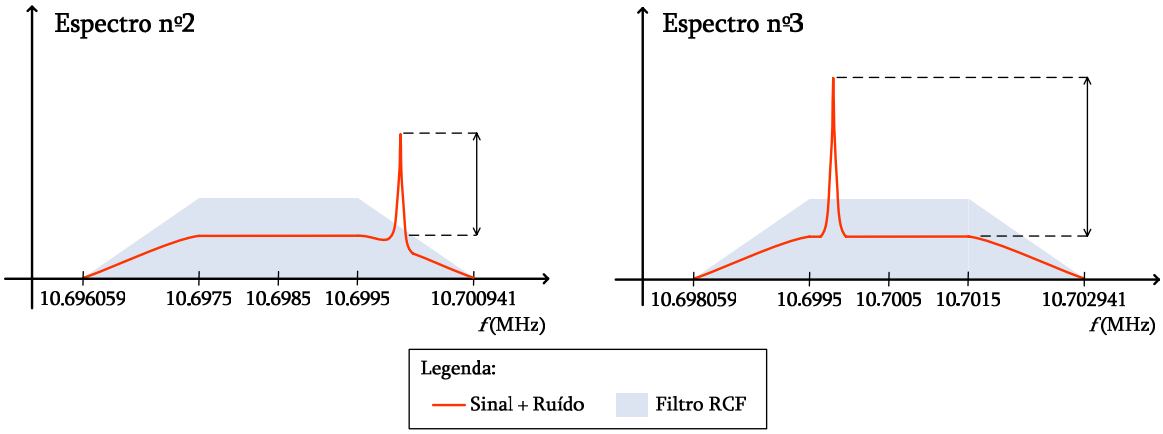

Figura 5.11 – Ilustração do problema na estimação de frequência.

Se o ruído fosse *flat*, tal como aconteceria numa situação real com o sinal proveniente do satélite, seria de esperar que a SNR fosse igual em ambos os casos já que o ganho do filtro afectaria igualmente o sinal e o ruído. No entanto isto não se verifica com o ruído gerado pelo equipamento utilizado no teste, sendo que ocasionalmente o ruído observado pode ser menor do que o esperado e logo a SNR toma valores respectivamente mais elevados ou vice-versa. De facto depois de repetir este teste várias vezes, verificou-se que em algumas ocasiões o espectro era escolhido correctamente.

Uma solução mais fiável para o varrimento consiste então em limitar o valor para o incremento de frequência, a um máximo correspondente à largura de banda a 3dB do filtro RCF. Desta forma garante-se que, qualquer que seja a frequência do sinal de entrada, existe sempre pelo menos um espectro em que o sinal está contido na banda de passagem deste filtro. Os espectros adjacentes apresentariam o sinal contido nos flancos ascendente ou descendente do filtro RCF, pelo que seria necessário descartá-los para a discriminação do espectro principal.

Para os testes finais, optou-se por efectuar o varrimento numa menor largura de banda que fosse aproximadamente igual à largura de banda do ruído (cerca de 5kHz). Relativamente ao valor do incremento de frequência e unicamente para efeitos de teste, optou-se por usar um valor inferior e correspondente a metade da largura de banda de passagem do filtro RCF. Desta forma existiriam sempre dois espectros em que o sinal

estaria contido na banda de passagem do filtro RCF, sendo que apenas um deles seria escolhido. Os novos parâmetros utilizados foram os seguintes:

- Frequência de varrimento inicial: 10.6975MHz;
- Incremento de frequência: 1kHz;
- Número de espectros: 5;
- Número de amostras para análise espectral: 512.

Repetiu-se então o teste para várias frequências e várias amplitudes do sinal de entrada. Os resultados obtidos e apresentados na Tabela 5.2, estavam finalmente de acordo com os objectivos pretendidos para este módulo de *software*. Além das frequências estarem a ser estimadas correctamente, os espectros escolhidos como principais também foram sempre os espectros correctos.

Atenuação (dB)	Frequência do Sinal de Entrada (MHz)	Frequência Estimada (MHz)	Potência de Sinal ($\times 10^6$)	Potência de Ruído ($\times 10^6$)	SNR (dB)	CNR (dB/Hz)	Densidade Espectral de Ruído
0	10.697700	10.697717	83561259.27	173454.23	26.83	59.87	334.40
	10.698800	10.698815	84643795.92	14380.79	37.70	59.09	399.89
	10.699700	10.699717	83141609.82	200262.48	26.18	55.90	832.70
	10.700800	10.700815	84266335.30	13694.10	37.89	57.58	564.75
3	10.697700	10.697716	42520792.25	68143.78	27.95	55.58	456.16
	10.698800	10.698815	42471648.40	17534.58	33.84	54.42	589.81
	10.699700	10.699717	42379961.04	117945.81	25.55	53.48	744.36
	10.700800	10.700815	42596905.04	15572.35	34.37	54.53	579.01
6	10.697700	10.697712	21845609.90	38090,62	27.59	52.66	464.13
	10.698800	10.698813	21721341.20	12376.07	32.44	51.92	539.29
	10.699700	10.699712	21363968.55	42034.61	27.06	50.40	764.48
	10.700800	10.700815	21024848.92	12965.20	32.10	51.89	528.57
9	10.697700	10.697715	10846209.21	25108.80	26.35	49.12	526.61
	10.698800	10.698815	10567650.03	11694.11	29.56	48.38	599.94
	10.699700	10.699716	10967341.51	34764.54	24.99	48.30	644.94
	10.700800	10.700815	10878472.42	12863.70	29.27	48.86	556.14
12	10.697700	10.697714	5485248.84	12379.06	26.47	46.09	540.73
	10.698800	10.698813	5453712.64	14612.76	25.72	46.39	504.77
	10.699700	10.699715	5397666.52	17046.79	25.01	45.96	553.42
	10.700800	10.700815	5571601.37	11328.02	26.92	46.03	556.95
15	10.697700	10.697711	2689230.62	14157.28	22.79	43.93	461.75
	10.698800	10.698814	2613980.70	13875.50	22.75	44.06	436.28
	10.699700	10.699712	2538579.91	13592.29	22.71	43.48	485.90
	10.700800	10.700813	2665895.01	11194.36	23.77	43.89	457.39
18	10.697700	10.697716	1304604.41	13435.71	19.87	41.80	406.92
	10.698800	10.698815	1293217.07	11527.62	20.50	41.10	468.33

Atenuação (dB)	Frequência do Sinal de Entrada (MHz)	Frequência Estimada (MHz)	Potência de Sinal ($\times 10^6$)	Potência de Ruído ($\times 10^6$)	SNR (dB)	CNR (dB/Hz)	Densidade Espectral de Ruído
	10.699700	10.699716	1177540.17	12096.38	19.88	41.66	389.42
	10.700800	10.700815	1270864.30	12180.90	20.18	41.25	454.20
21	10.697700	10.697715	643948.75	13835.72	16.68	39.75	387.82
	10.698800	10.698813	725495.13	11411.80	18.03	40.13	383.55
	10.699700	10.699714	643069.08	11029.34	17.66	39.82	390.25
	10.700800	10.700813	644742.26	12497.67	17.13	39.39	425.94
24	10.697700	10.697713	365501.25	10525.15	15.41	39.30	307.92
	10.698800	10.698814	360629.50	10934.90	15.18	39.44	317.91
	10.699700	10.699715	359440.65	13633.23	14.21	38.63	375.04
	10.700800	10.700815	339033.17	11643.21	14.64	38.87	352.85
27	10.697700	10.697716	172475.26	12252.61	11.49	39.16	257.54
	10.698800	10.698814	202746.82	10299.67	12.94	38.90	263.37
	10.699700	10.699715	168625.19	9933.72	12.30	38.04	297.05
	10.700800	10.700812	160654.35	10746.53	11.75	38.84	270.27
30	10.697700	10.697717	61323.67	10507.53	7.66	38.86	215.43
	10.698800	10.698814	103778.57	10677.24	9.88	38.67	249.58
	10.699700	10.699718	77977.96	9718.51	9.04	37.20	269.94
	10.700800	10.700817	107190.22	11814.22	9.58	38.31	281.90

Tabela 5.2 – Resultados dos testes finais da estimação da frequência do sinal de entrada.

Como se pode observar, as frequências estimadas estão bastante próximos das frequências do sinal de entrada correspondentes. As diferenças de frequência observadas são suficientemente pequenas para permitir à PLL adquirir o sincronismo rapidamente.

Os valores de SNR obtidos variam muito para atenuações mais reduzidas, o que poderá dever-se ao facto destes valores serem estimados no domínio das frequências e condicionados pelo fenómeno de Gibbs. Ao examinar em pormenor o espectro do sinal de entrada (ver Figura 5.12), é possível constatar o espalhamento da potência do sinal devido ao fenómeno de Gibbs, tal como se referiu na secção 3.1.3. Como seria de esperar, quando a atenuação do sinal aumenta, o patamar de ruído sobrepõe-se ao *leakage* e os efeitos adversos do fenómeno de Gibbs reduzem-se, pelo que os valores são mais estáveis quando a atenuação é elevada.

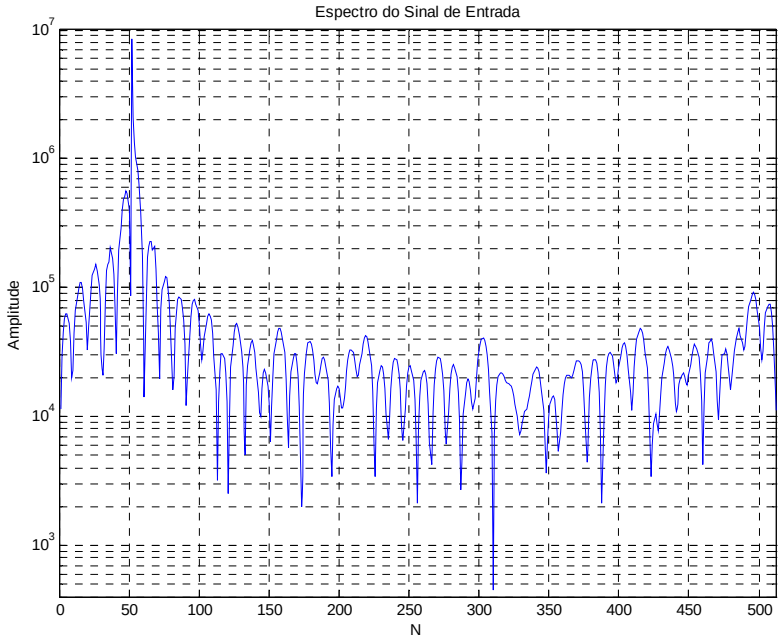

Figura 5.12 – *Leakage* num espectro do sinal de entrada calculado por FFT (Δf = 500Hz).

No entanto, os valores obtidos no domínio dos tempos para a CNR e densidade espectral de potência de ruído também apresentam algumas incoerências, que se poderão dever ao facto do ruído não estar a ser gerado tal como se pretendia. Embora os valores de CNR sejam bem estimados para atenuações reduzidas, o mesmo não acontece para valores mais elevados. Por outro lado, também se esperava que os valores obtidos para a densidade espectral de potência de ruído fossem aproximadamente constantes, ou que não apresentassem tão grandes variações como as que se verificam. Para melhorar a estimação destes valores seria necessário aumentar o número de pontos recolhidos para o cálculo da FFT, sendo esta uma questão que deverá se considerada futuramente.

De referir ainda que o tempo necessário para completar o processo de estimação da frequência do sinal de entrada é da ordem de 1s.

5.2. Cadeia Secundária de Filtragem e Decimação

Antes de testar o código para a implementação deste módulo, foi necessário averiguar se o tempo do processamento mais exigente, poderia exceder o tempo disponível entre a recepção de amostras consecutivas, tal como se referiu na secção 3.2. Se isto se verificar, a consequente perda de amostras constituiria um problema grave e incontornável para o sistema.

Seleccionaram-se então as instruções relativas ao trecho computacionalmente mais exigente que é o despoletado a partir do instante em que se recebe a última amostra de cada grupo de *MTOTAL* amostras. Este processamento corresponde portanto aos cálculos do filtro FIR1 seguidos dos cálculos do filtro FIR2.

Para medir o tempo de processamento em número de ciclos de relógio da DSP, recorreu-se à função *Profiler Clock* do *Code Composer Studio* e executou-se o código da seguinte forma:

1. Activa-se o relógio através da opção *Profiler/Enable Clock* e de seguida activa-se a opção *Profiler/View Clock* para visualizar de imediato a janela do relógio;
2. Colocam-se dois *breakpoints* na primeira e última instrução do trecho de código que se pretende analisar;
3. Compila-se o código e carrega-se o ficheiro executável para a DSP;
4. Corre-se o programa, sendo que a sua execução fica parada no primeiro *breakpoint*. Faz-se o *reset* ao relógio e corre-se novamente o programa, que é executado até ao segundo *breakpoint*. O relógio passa a indicar então o número de ciclos de relógio da DSP necessários para executar o trecho de código seleccionado.

Como a taxa de amostragem é de 4882S/s, o tempo máximo disponível para o processamento é dado por:

$$T_{\max} = \frac{1}{4882} = 0.2048 ms\,, \tag{5.2}$$

o que em termos de ciclos de relógio de 150MHz equivale a aproximadamente a 30725 ciclos.

No entanto, os primeiros resultados obtidos apontavam para cerca de 80000 ciclos de relógio, o que ultrapassaria largamente o valor máximo referido anteriormente. Assim, apesar de toda a preocupação com a carga computacional na implementação dos filtros, foi necessário recorrer a optimização de *software* e é aqui que se revelam todas as potencialidades das DSPs. Mesmo sem uma optimização exaustiva, foi possível reduzir drasticamente o tempo de processamento.

De acordo com o que se referiu na secção 4.9, activou-se o nível máximo de optimização (*-o3*) mais focado na velocidade e em deterimento do tamanho do código (*no -ms*). Seleccionaram-se então as seguintes opções no compilador:

- *-o3* : nível máximo de optimização do compilador;
- *-gp* : gera informação de *debug* (*function profile debug*);
- *-k* : mantém os ficheiros de *assembly* "*.asm*" para posterior análise da informação do *feedback* do compilador;
- *-on2* : gera ficheiros com informação verbosa sobre a optimização (*optimization info file: verbose*);
- *-q* : *no banners*.

Para facilitar o acesso à memória, adicionaram-se também as directivas *#PRAGMA DATA_ALIGN()* aos *arrays* dos *buffers* e dos coeficientes dos filtros FIR, para um alinhamento destes dados na memória em endereços múltiplos de 4 bytes (que corresponde ao tamanho das variáveis de tipo *float*).

De imediato foi possível reduzir o tempo de processamento para cerca de 30000 ciclos, o que apesar de ser uma redução bastante considerável estava ainda não seria a necessária para garantir uma boa margem. Analisou-se então o ficheiro "*conf.asm*", onde se verificou que apesar do *software pipelining* estar activo, aparecia a seguinte mensagem de erro antes dos ciclos relativos aos cálculos dos filtros FIR1 e FIR2:

```
;*----------------------------------------------------------------------------*
;*  SOFTWARE PIPELINE INFORMATION
;*     Disqualified loop: loop contains a call
;*----------------------------------------------------------------------------*
```

De acordo com [10, pág. 58] e com [9, cap. 2, pág. 55], estas mensagens de erro acusavam a desqualificação dos referidos ciclos para o *software pipelining* devido à existência de chamadas a rotinas que não poderiam ser incluidas num ciclo *pipelined*, nomeadamente a chamada a rotinas de suporte em tempo real. Neste caso o problema estava na utilização da operação "*%*" que invocava a rotina *_remi*.

Efectuaram-se então as alterações necessárias ao código destes ciclos e compilou-se novamente o programa. Analisando novamente o ficheiro "*conf.asm*", foi possível verificar então que os ciclos já estavam a ser *software-pipelined*, já que antes de cada um deles era possível observar mensagens contendo informação relativa ao *software pipelining*. No ciclo relativo aos cálculos do filtro FIR1 aparecia a seguinte mensagem:

```
;*----------------------------------------------------------------------------*
;*  SOFTWARE PIPELINE INFORMATION
;*
;*     Loop source line                 : 952
;*     Loop opening brace source line   : 953
;*     Loop closing brace source line   : 969
;*     Known Minimum Trip Count         : 120
;*     Known Maximum Trip Count         : 120
;*     Known Max Trip Count Factor      : 120
;*     Loop Carried Dependency Bound(^) : 4
;*     Unpartitioned Resource Bound     : 3
;*     Partitioned Resource Bound(*)    : 3
;*     Resource Partition:
;*                               A-side   B-side
;*     .L units                     1       3*
;*     .S units                     2       1
;*     .D units                     2       2
;*     .M units                     0       2
;*     .X cross paths               0       2
;*     .T address paths             1       2
;*     Long read paths              0       0
;*     Long write paths             0       0
;*     Logical  ops (.LS)           0       0    (.L or .S unit)
;*     Addition ops (.LSD)          2       3    (.L or .S or .D unit)
```

```
;*      Bound(.L .S .LS)              2       2
;*      Bound(.L .S .D .LS .LSD)    3*      3*
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 4  Schedule found with 5 iterations in parallel
;*      done
;*
;*      Epilog not entirely removed
;*      Collapsed epilog stages     : 1
;*
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 0 bytes
;*
;*      For further improvement on this loop, try option -mh4
;*
;*      Minimum safe trip count     : 3
;*----------------------------------------------------------------------------*
```

Observando a mensagem anterior constata-se que o compilador aconselha a utilização da opção *–mh4* para um melhor desempenho. A opção *–mh<n>* permite a execução especulativa, em que a quantidade apropriada de *data memory padding* que assegura a correcta execução do ciclo é dada por *n*.

Seleccionou-se então esta opção e no final da optimização, foi possível obter um tempo de processamento inferior a 4000 ciclos e portanto muito mais curto do que o obtido anteriormente.

O teste para a depuração deste módulo consistiu basicamente na recepção de 20000+1880 amostras com a malha aberta, que seriam filtradas e decimadas pelos filtros FIR1 e FIR2. Por cada $M1$ (neste caso 40) das amostras recebidas, seria calculada uma amostra à saída do filtro FIR1. As 500 amostras resultantes seriam então armazenadas para serem posteriormente analisadas e comparadas com as amostras recebidas. Analogamente, por cada $M2$ (neste caso 5) das amostras calculadas pelo filtro FIR1, seria calculada uma amostra no filtro FIR2, sendo que as 100 amostras resultantes seriam também armazenadas e comparadas com as amostras recebidas. Para comparar os sinais obtidos à saída de cada um dos filtros FIR com os sinais esperados e obtidos com o *Matlab*, correu-se o *m-file* presente na secção B.

De referir que, dado o elevado factor de decimação, para se armazenar um número suficiente de amostras à saída dos filtros FIR, seria necessário receber um número elevado de amostras à saída do detector de fase. Para poder comparar as amostras à saída dos filtros com as amostras à entrada, seria necessário armazenar também estas últimas. No entanto, não é possível armazenar em memória tais quantidades de dados, pelo que se optou por armazenar apenas uma pequena fracção das amostras recebidas (5000).

Depois da primeira execução deste teste, verificou-se que o sinal filtrado e decimado, embora tivesse um comportamento próximo das sinusóides do sinal de

entrada, estava muito longe de corresponder ao esperado. Verificou-se mais tarde que isto se devia à omissão de uma instrução no código, relativa à gestão dos pontos de retirada dos *buffers* circulares. Apenas 1 em cada 3 ou 10 amostras estariam a ser calculadas correctamente, respectivamente no filtro FIR1 e FIR2.

Depois de corrigido o erro referido anteriormente, repetiu-se a execução do teste para duas diferenças de frequência (Δf) entre o NCO e o sinal de entrada. Os resultados relativos aos filtros FIR1 estão representados na Figura 5.13, na Figura 5.14 e na Figura 5.15.

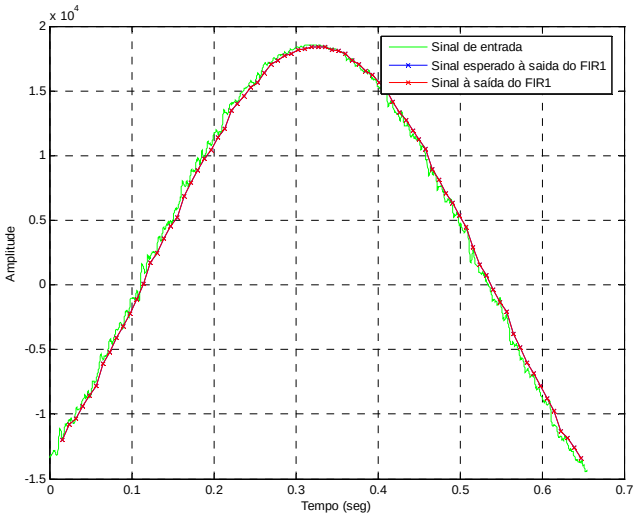

Figura 5.13 – Gráficos com os sinais à entrada e à saída do filtro FIR1 (Δf= 1.2Hz).

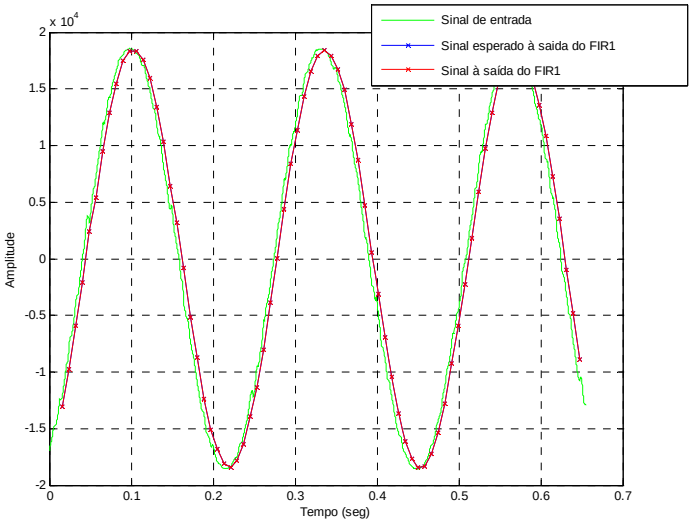

Figura 5.14 – Gráficos com os sinais à entrada e à saída do filtro FIR1 (Δf= 4.3Hz)

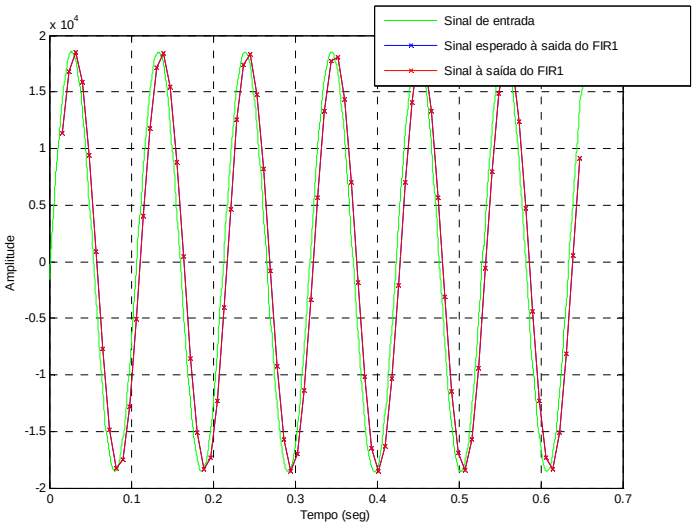

Figura 5.15 – Gráficos com os sinais à entrada e à saída do filtro FIR1 (Δf= 9.4Hz)

Nas figuras anteriores, a sobreposição do sinal esperado (a azul) e do sinal obtido (a vermelho) à saída do filtro FIR1 é praticamente perfeita, pelo que as amostras obtidas com a DSP são praticamente iguais às amostras calculadas recorrendo ao *Matlab*. Desta forma, para frequências inferiores à frequência de corte do filtro FIR2 e logo bastante inferiores à frequência de corte do filtro FIR1, o sinal obtido constitui uma boa aproximação do sinal à saída do detector de fase (a verde). Todavia deve-se referir ainda a existência de um atraso no sinal à saída do filtro FIR1 em relação ao sinal de entrada, tal como acontece em qualquer filtro analógico.

Na Figura 5.16, na Figura 5.17 e na Figura 5.18 estão representados os resultados relativos ao filtro FIR2.

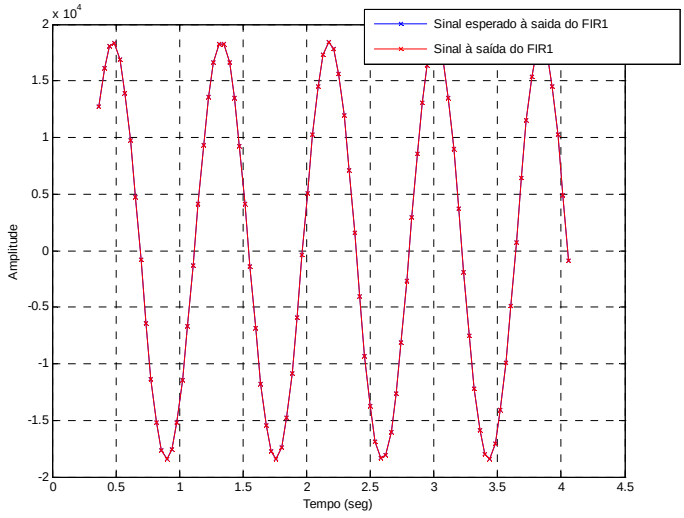

Figura 5.16 – Gráficos com os sinais à saída do filtro FIR2 (Δf= 1.2Hz).

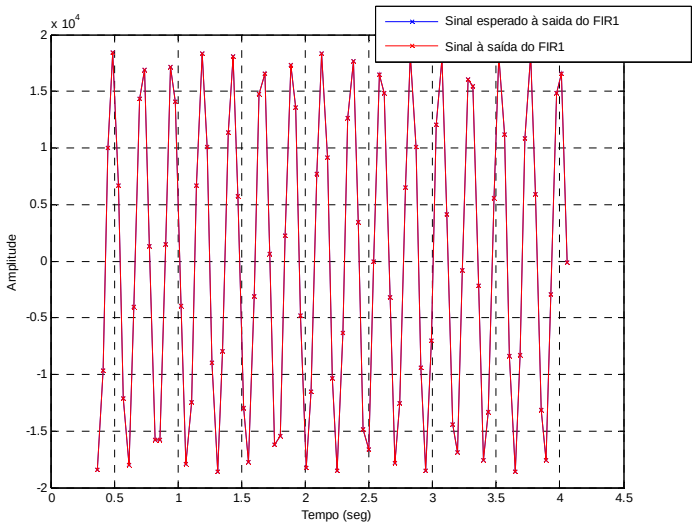

Figura 5.17 – Gráficos com os sinais à saída do filtro FIR2 (Δf= 4.3Hz).

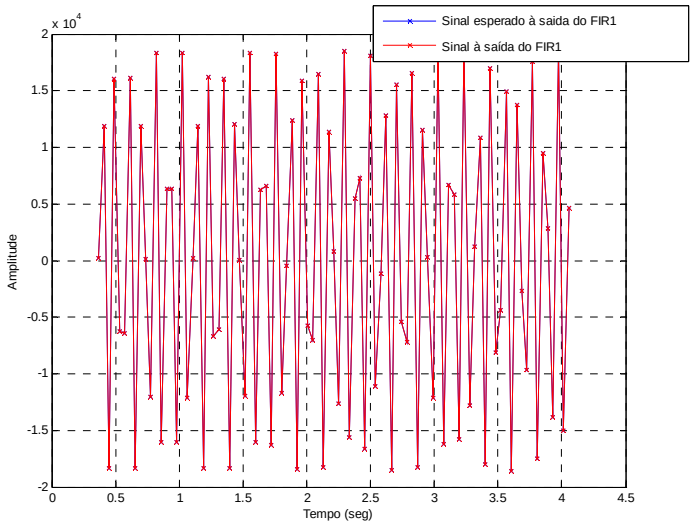

Figura 5.18 – Gráficos com os sinais à saída do filtro FIR2 (Δf= 194Hz).

Analogamente aos resultados do filtro anterior, o sinal obtido à saída do filtro FIR2 está de acordo com o sinal esperado e calculado através do *Matlab*. No entanto, ao contrário do que se observa no filtro FIR1, o sinal à saída do filtro FIR2 só constitui uma aproximação razoável do sinal de entrada para frequências bastante inferiores à frequência de corte deste filtro. O mesmo já não se verifica quando a frequência do sinal de entrada se aproxima deste valor, o que se deve à amostragem. Nestes casos a frequência do sinal de entrada é só ligeiramente superior ao dobro da frequência de amostragem neste filtro, o que permite apenas a existência de pouco mais que dois pontos por cada período do sinal de entrada. Tal como no filtro FIR1, verificou-se também a existência de um atraso no sinal obtido à saída filtro FIR2. Assim, estando os filtros FIR em cascata, o atraso total do sinal ao passar na cadeia secundária de filtragem e decimação é de aproximadamente 0.385ms.

Depois de verificar a concordância dos resultados obtidos com os esperados, foi possível confirmar a validade da solução utilizada para implementar os filtros FIR, tal como eram pretendidos para a cadeia secundária de filtragem e decimação.

5.3. Controlo Automático de Ganho (AGC)

Depois do executado o *debugging* para a cadeia secundária de filtragem e decimação, o passo seguinte foi testar a implementação do controlo automático de ganho.

O primeiro teste consistiu em arrancar a PLL e o AGC e adquirir cerca de 300 000 amostras em aproximadamente 1 minuto, diminuindo a amplitude do sinal no gerador em incrementos de 0.5dB a cada 4s (aproximadamente), desde 0dB até -9dB. Durante o teste, foram armazenadas as cerca de 1500 amostras das componentes I e Q actuadas pelo AGC, que estão representadas no gráfico da Figura 5.19.

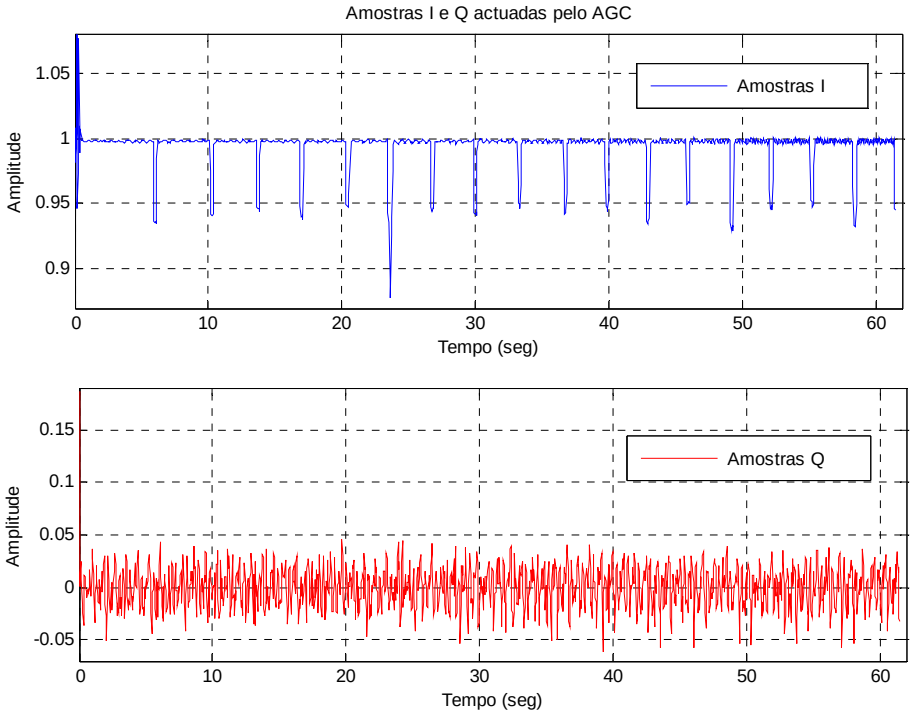

Figura 5.19 – Gráfico com as amostras das componentes I e Q actuadas pelo AGC.

Como se pode observar, apesar do AGC demorar algum tempo para responder às alterações de amplitude do sinal de entrada, este consegue contrariá-las, mantendo a amplitude do sinal na sua saída aproximadamente constante e igual a 1. Tal facto pode ser observado em maior detalhe na Figura 5.20. Verificou-se ainda que, tal como seria de esperar, quanto maior é a atenuação do sinal, menos constante é o sinal à saída do AGC.

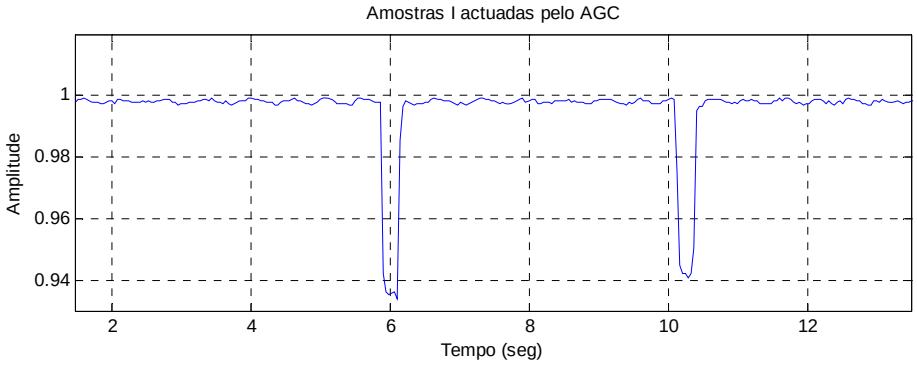

Figura 5.20 – Gráfico pormenorizado das amostras da componente I actuada pelo AGC.

Figura 5.21 – Atenuadores em cascata (*stepped attenuators*).

De seguida ligaram-se dois atenuadores em cascata (ver Figura 5.21) à saída RF do gerador de sinal, sendo que um deles gerava atenuações entre 0 e 50dB, com incrementos de 10dB, e o outro gerava atenuações mais finas com incrementos de 1dB entre 0 e 10dB. Repetiu-se o teste anterior por quatro vezes, atenuando o sinal em incrementos de 1dB desde 0 até 9dB, desde 10 até 19dB, desde 20 até 29dB e por último desde 30 até 39dB. Os resultados obtidos podem ser observados respectivamente na Figura 5.22, Figura 5.23, Figura 5.24 e Figura 5.25.

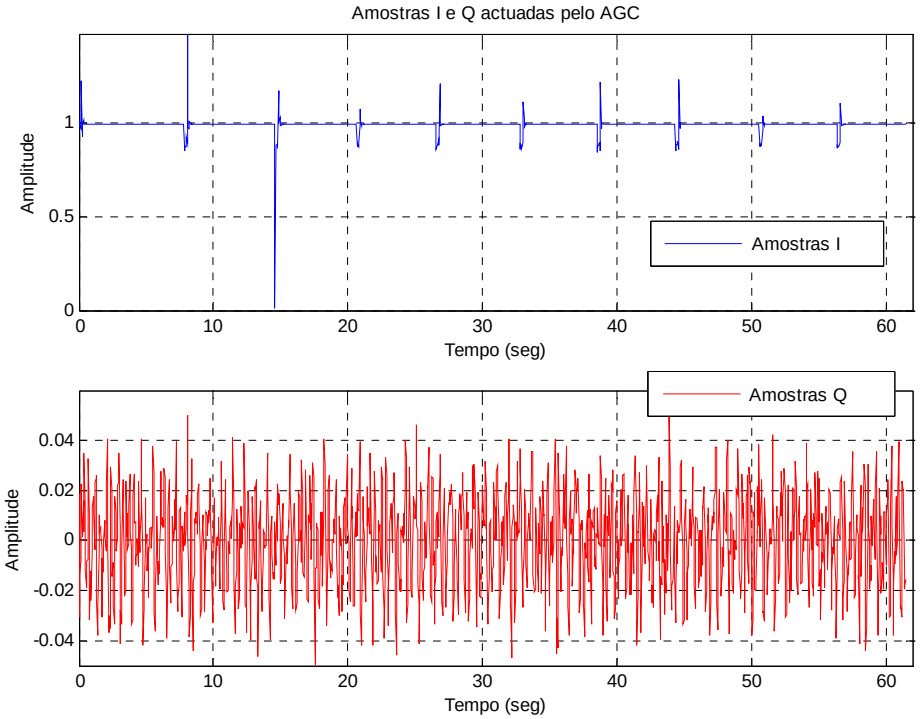

Figura 5.22 – Gráficos com a atenuação a variar entre 0 e 9dB, com incrementos de 1dB.

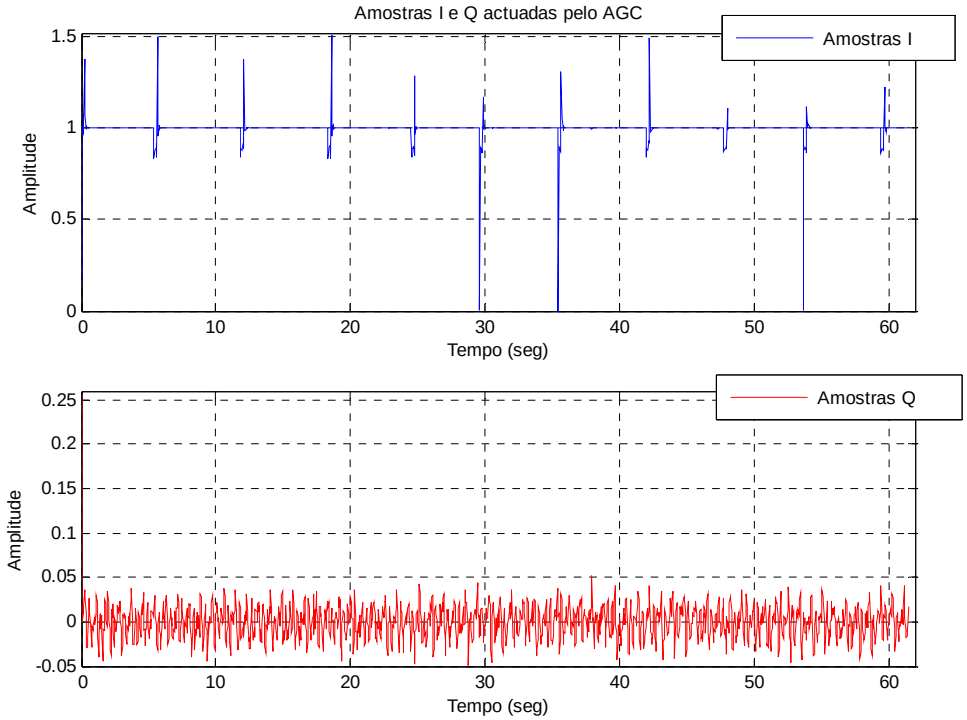

Figura 5.23 – Gráficos com a atenuação a variar entre 10 e 19dB, com incrementos de 1dB.

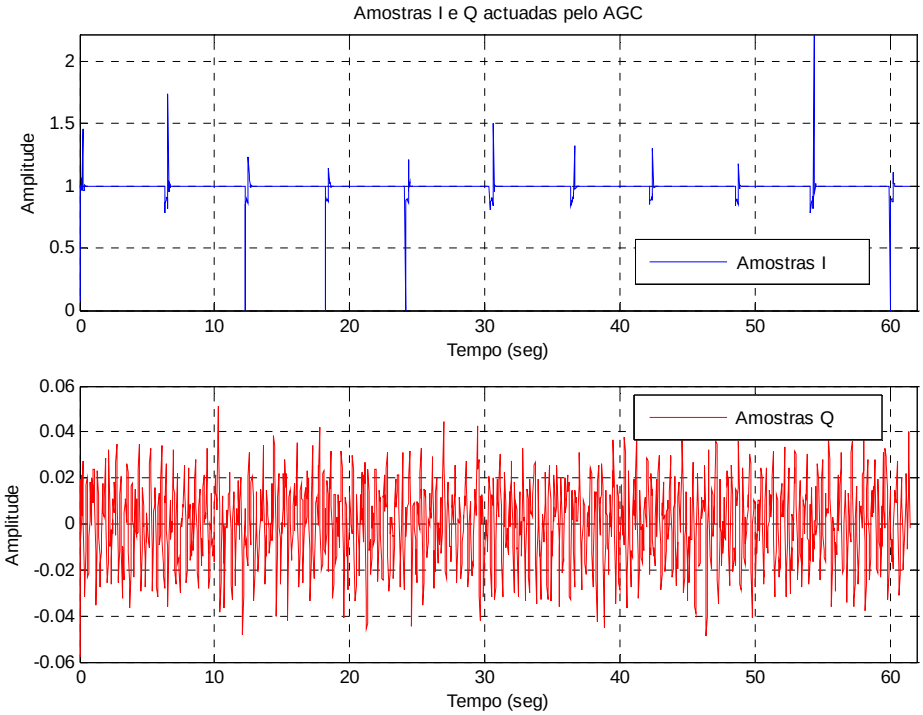

Figura 5.24 – Gráficos com a atenuação a variar entre 20 e 29dB, com incrementos de 1dB.

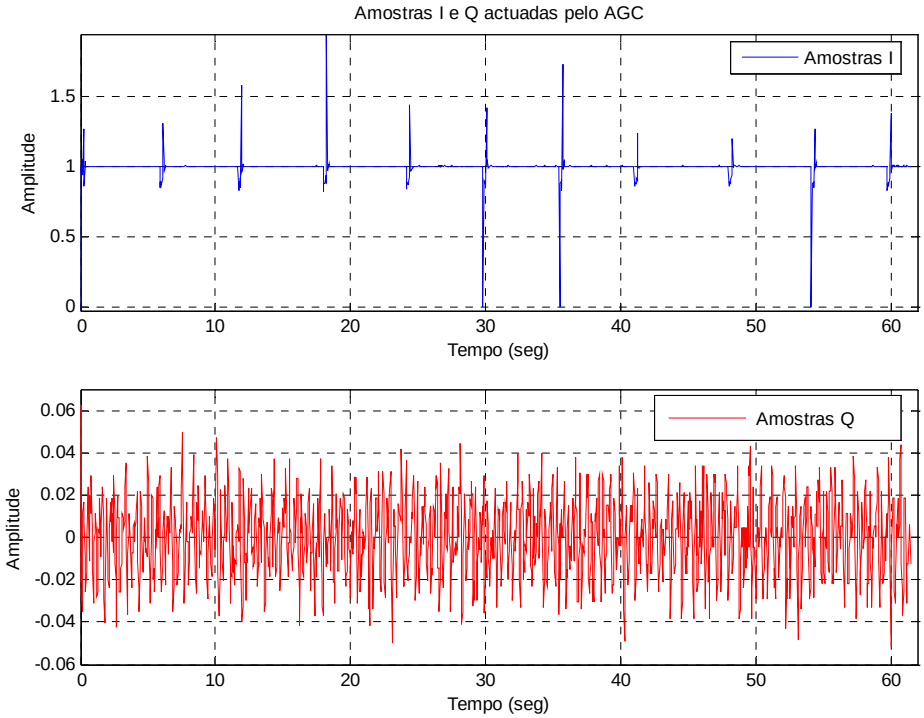

Figura 5.25 – Gráficos com a atenuação a variar entre 30 e 39dB, com incrementos de 1dB.

Como se pode observar nas figuras anteriores, as amostras I actuadas pelo AGC apresentam um valor praticamente constante e rondando a unidade, durante a maior parte do tempo entre variações consecutivas de amplitude. No entanto, ao contrário do que se esperaria, estas variações da amplitude causam variações muito bruscas e

indesejadas sobre a componente I actuada pelo AGC. A causa deste problema reside no mecanismo dos atenuadores em cascata, em que a rodagem dos botões até à posição desejada leva à abertura breve da cadeia de sinal (abre antes de fechar), o que não representa de forma alguma a variação suave do sinal numa situação real.

Executou-se novamente o código do teste anterior, mas armazenando os valores da frequência configurados no NCO a cada 24.41s, seguindo a mesma cadência da saída do filtro FIR2, e desligando o gerador após alguns segundos. O objectivo seria analisar o comportamento do AGC após a perda de sinal. Os resultados estão representados na Figura 5.26.

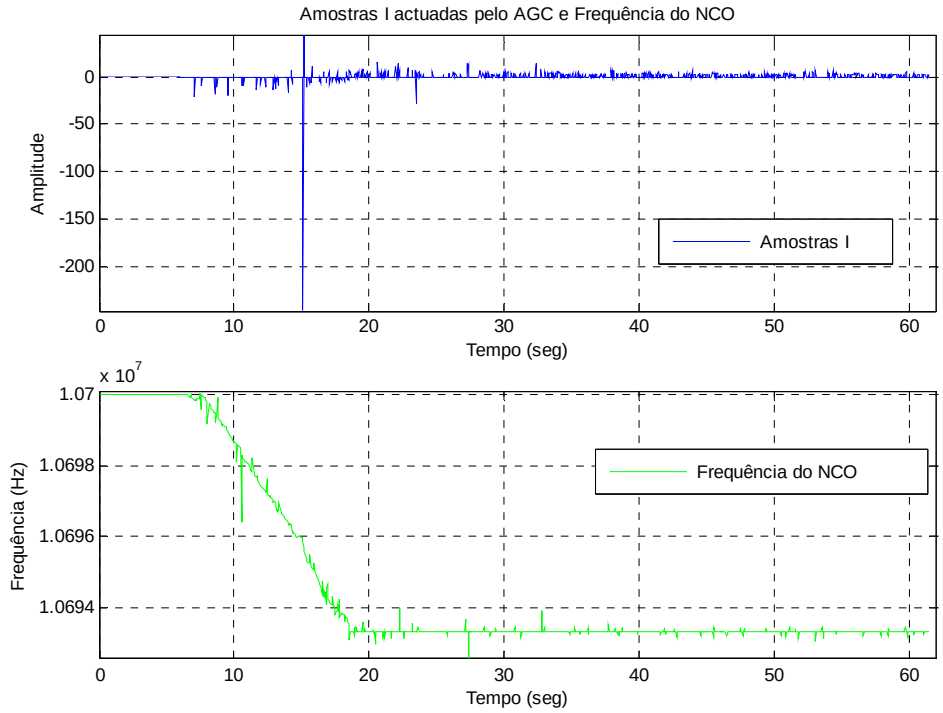

Figura 5.26 – Gráficos com a componente I actuada pelo AGC e os valores de frequência do NCO.

Como se pode observar na Figura 5.26, enquanto o gerador está ligado, a componente I actuada pelo AGC mantém-se praticamente constante a rondar a unidade e a frequência mantém-se também praticamente constante nos 10.7MHz. No entanto, depois de desligar o gerador, em vez da frequência se manter constante, esta sofre um desvio enorme de cerca de quase 5kHz. Este desvio deve-se ao facto da componente I actuada pelo AGC não se manter constante e próxima de zero.

Figura 5.27 – Gerador de funções/formas de onda da *Wavetek* – modelo 180.

Devido à limitação dos atenuadores em cascata utilizados nestes testes, optou-se por proceder a novos testes utilizando um outro gerador de funções/formas de onda da *Wavetek* (ver Figura 5.27), para gerar um sinal de baixa frequência como fonte para uma modulação externa no gerador de sinal da *Marconi Instruments*, tal como mostra o esquema da Figura 5.28.

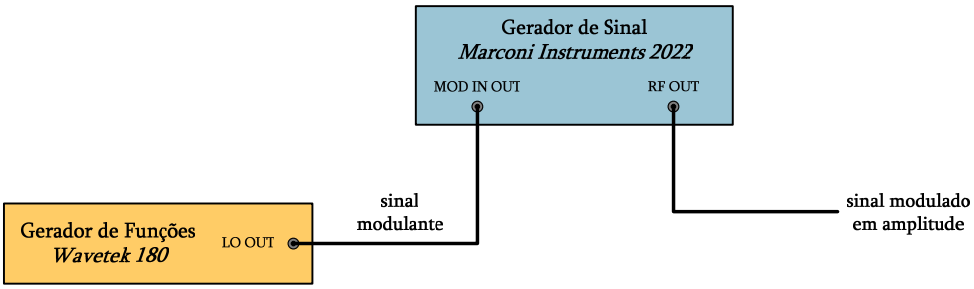

Figura 5.28 – Esquema do equipamento utilizado para modular o sinal em amplitude.

Para observar o sinal modulado e verificar os níveis máximo e mínimo de amplitude correspondentes, recorreu-se um osciloscópio da Figura 5.29.

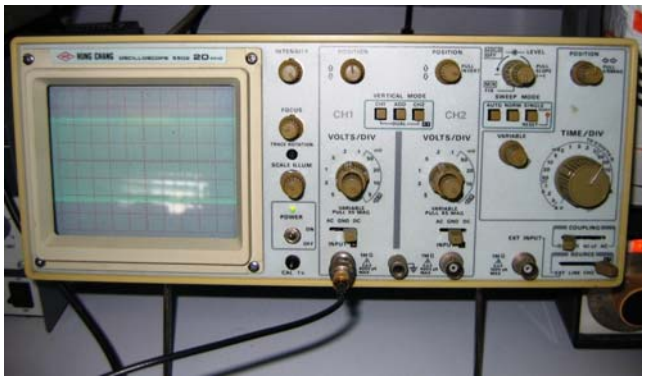

Figura 5.29 – Osciloscópio da *Hung Chang* – modelo 5502.

Desta forma, o sinal modulado poderia simular o sinal de entrada de uma forma muito mais realista, com variações mais suaves de atenuação, uma vez que não são esperadas variações tão bruscas como as que foram observadas nos testes anteriores.

Com um sinal modulante próximo dos 0.01Hz, executou-se novamente o código de teste, armazenando adicionalmente as amostras da componente I à saída do filtro FIR2. Os resultados obtidos podem ser observados na Figura 5.30.

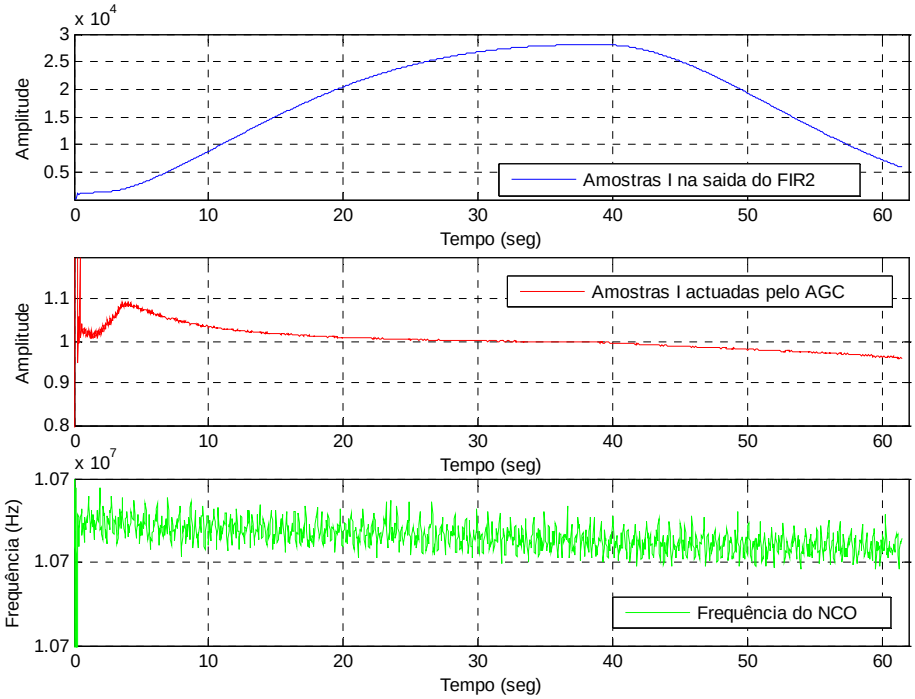

Figura 5.30 – Gráficos com os resultados obtidos com uma frequência modulante de 0.01Hz.

Como se pode observar, a componente I actuada pelo AGC mantém-se muito próxima de 1 apesar da amplitude do sinal variar bastante. No entanto verificou-se uma oscilação inicial que se poderá dever ao facto do sinal modulante do gerador ter uma irregularidade para a amplitude mais baixa. Repare-se no entanto que existem periodos em que há um desvio moderado do valor unitário. Isto parece ser devido a dificuldades em a malha responder com adequada agilidade à variação de amplitude do sinal de entrada. Isto levou, embora os efeitos da "imperfeição sejam pouco relevantes para a malha, à derivação da taxa de variação do sinal na simulação.

Conhecendo a amplitude máxima $A_{\max}$ e a amplitude mínima $A_{\min}$ do sinal modulado, é possível obter uma expressão para a amplitude deste sinal ao longo do tempo:

$$A = A_o\left(1+\frac{A_{\max}-A_{\min}}{A_{\max}+A_{\min}}\cos\omega_m t\right) = A_o\left(1+0.93\cos\omega_m t\right), \qquad \textbf{(5.3)}$$

onde A_o é a amplitude média do sinal modulado, ω_m é a frequência do sinal modulante e 0.93 o valor que se adequa ao teste realizado neste caso.

Em termos de dB a equação (5.3) pode ser reescrita como:

$$\begin{aligned} A_{dB} &= 20\log_{10} A_o + 20\log_{10}\left(1+0.93\cos\omega_m t\right) \Leftrightarrow \\ \Leftrightarrow A_{(dB)} - A_{o(dB)} &= 8.686\ln\left(1+0.93\cos\omega_m t\right) \end{aligned} \quad . \tag{5.4}$$

Derivando a equação anterior em ordem ao tempo é possível determinar a variação em dB da amplitude do sinal mudulado:

$$\frac{dA_{(dB)}}{dt} = -8.686\frac{0.93\omega_m \sin\omega_m t}{1+0.93\cos\omega_m t}. \tag{5.5}$$

Na Figura 5.31 está representada graficamente a equação (5.5) para a frequência do sinal modulante no teste anterior, onde se pode observar que o máximo da variação da amplitude do sinal modulado é de cerca de 1.38dB/s. Este é um valor bem mais realista que as mudanças abruptas do *stepped attenuator* contudo ainda bem acima da taxa de variação de atenuação de um canal troposférico em relação ao caso da simulação.

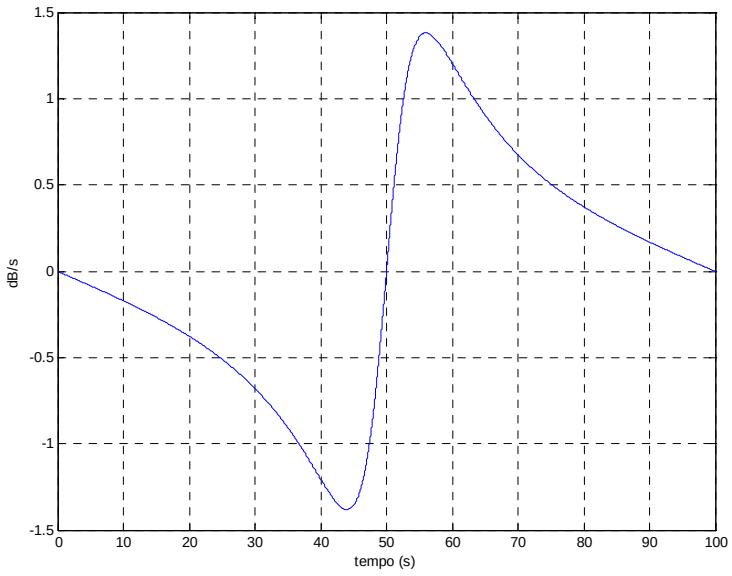

Figura 5.31 – Gráfico da variação da amplitude do sinal modulado.

5.4. Indicação de Sincronismo

O passo seguinte foi o *debugging* do módulo de indicação de sincronismo. O teste executado consistiu em arrancar com a execução da PLL e do AGC com o sinal de entrada com ruído e recolher as componentes I e Q, a frequência do NCO e o valor da *flag* de indicação sincronismo ao longo do tempo, até a recepção de amostras estar concluída.

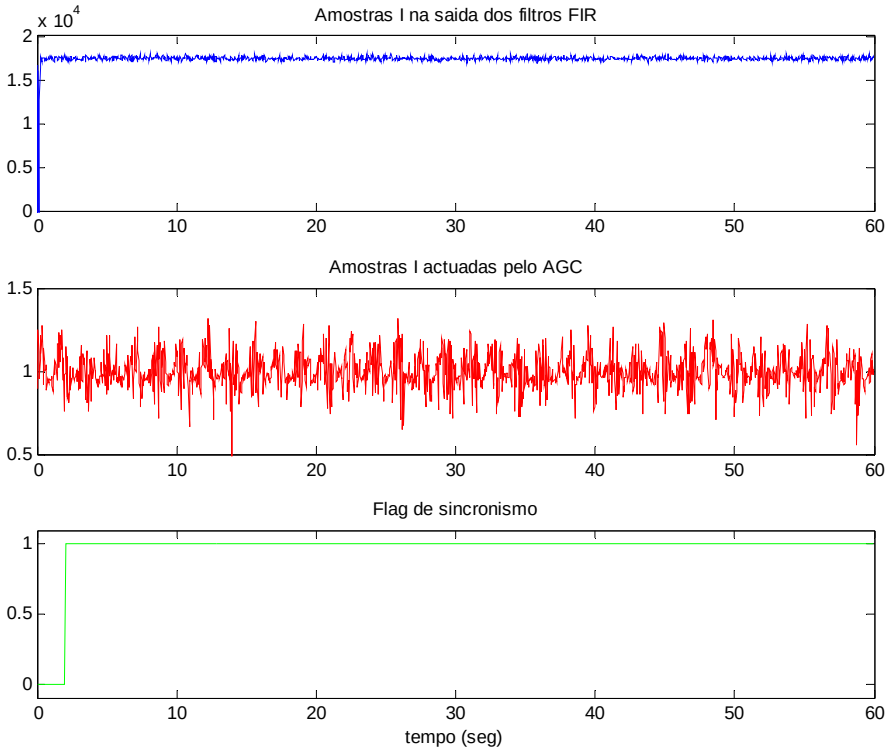

Figura 5.32 – Gráficos com a indicação da malha em sincronismo.

Os resultados obtidos estão representados na Figura 5.32 e estão de acordo com o esperado já que após aproximadamente 2s, a *flag* de indicação sincronismo passa do estado 0 (*unlocked*) para o estado 1 (*locked*), indicando que a malha está em sincronismo. Este período corresponde à soma do tempo dado pelo sistema para a malha entrar em sincronismo e do tempo de integração para obtenção da indicação de sincronismo (ambos de cerca de 1s).

5.5. Estimação de CNR e "Congelamento" do NCO

O passo seguinte foi o *debugging* do módulo de indicação de sincronismo. O teste executado consistiu em arrancar com a execução da PLL e do AGC com o sinal de entrada com ruído e recolher as componentes I e Q, a frequência do NCO e o valor da *flag* de indicação sincronismo ao longo do tempo, até a recepção de amostras estar concluída. De referir ainda que, nos primeiros testes o NCO era "congelado" ao valor de frequência correspondente ao sinal de entrada. Desta forma, tinha-se a certeza que a malha poderia recuperar o sincronismo instantaneamente.

Depois de executar vários testes, verificou-se que o "congelamento" do NCO umas vezes tinha sucesso e a malha readquiria o sincronismo, mas outras vezes o mesmo não sucedia. Na Figura 5.33 estão representados alguns resultados que exemplificam este problema no "congelamento" do NCO. O sinal utilizado neste teste era gerado com ruído e modulado tal como se descreveu na secção 5.1. Para diminuir ainda mais a

amplitude do sinal e obter valores mais reduzidos de CNR, intercalou-se um atenuador fixo de 10dB entre o gerador de sinal e o *power splitter*.

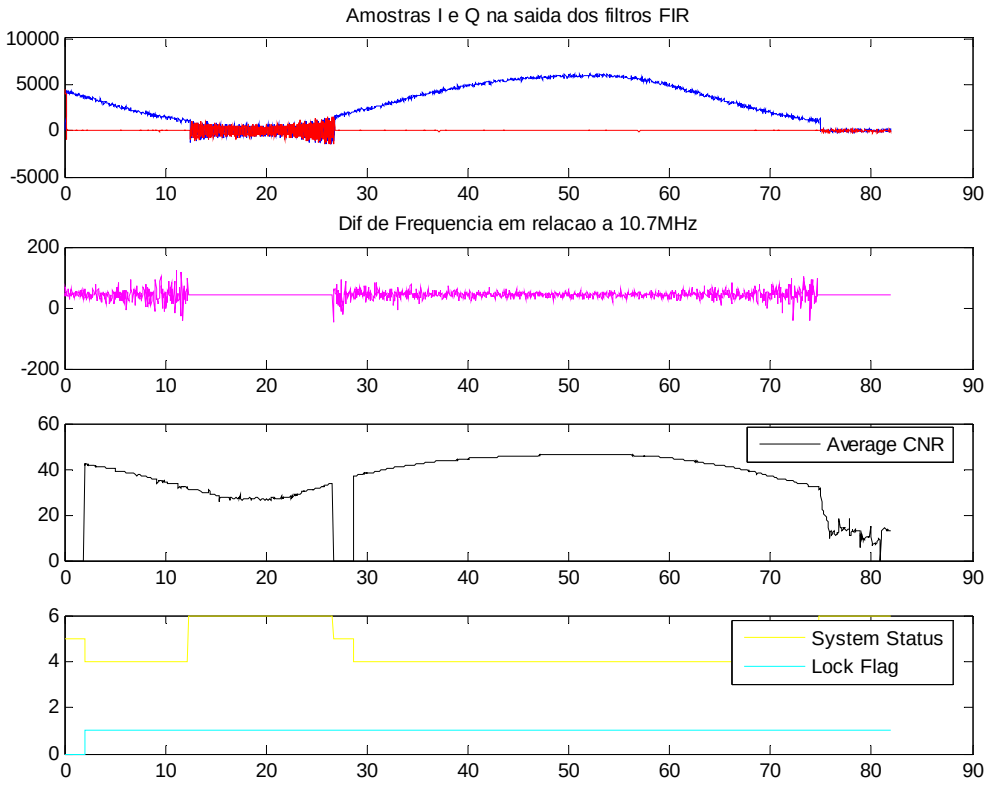

Figura 5.33 – Resultados dos testes ao "congelamento" do NCO.

Como se pode observar, da primeira vez em que a amplitude do sinal desce abaixo de um determinado valor, ao qual correspondem valores de CNR muito baixos, o NCO é "congelado" a uma frequência constante (a malha está aberto). Quando a amplitude do sinal volta a subir, a malha é fechada novamente e o sincronismo é adquirido novamente. No entanto, quando o NCO é "congelado" pela segunda vez, o sinal desaparece pelo que o NCO não é "descongelado" e a malha não recupera o sincronismo.

No primeiro "congelamento", as componentes I e Q apresentam o comportamento sinusoidal característico da malha aberto, mas com uma frequência que corresponderá à diferença entre a frequência do sinal de entrada e a frequência à qual o NCO é "congelado". No entanto, seria de esperar que esta frequência não fosse tão elevada, uma vez que o NCO estava a ser (aparentemente) "congelado" à frequência do sinal de entrada. Por outro lado, no segundo congelamento as componentes I e Q tomam valores nulos, o que poderia indicar uma diferençã significativa entre ambas.

Os valores de CNR estão também a ser estimados correctamente, uma vez que esta deveria oscilar entre 45 e 26dB/Hz, devido à modulação (que introduz atenuações até cerca de 19dB) e ao atenuador fixo de 10dB. De referir ainda que, a estimação de CNR não era efectuada enquanto o sistema procedia ao teste de sincronismo. No entanto, isto

devia-se ao facto da rotina que estima a CNR não ser evocada durante este procedimento, pelo que este problema foi facilmente corrigido para a realização dos teste seguintes.

Verificou-se ainda que a variação do estado do sistema ocorre de acordo com o esperado, uma vez que quando a CNR desce abaixo de um determinado limite minimo (neste caso 30dB/Hz), a *flag* toma o valor "6" correspondente ao "congelamento" do NCO. Após alguns segundos, auando a CNR volta a subir e ultrapassa um determinado valor (também 30dB/Hz), o sistema considera que o sinal recuperou e então procede ao teste do sincronismo, sendo que a *flag* toma o valor "5" durante 2s (exactamente o tempo necessário para este procedimento). Como a malha readquiriu o sincronismo, o teste foi efectuado com sucesso e a *flag* do estado do sistema retoma o valor "4" e o sistema procede à execução normal da PLL.

Mais tarde, tal como se referiu anteriormente, descobriu-se que o problema anterior estava relacionado com o valor da frequência a que o NCO era "congelado", já que este não correspondia ao valor esperado e que era o valor imprimido nos testes efectuados. Verificou-se então que na iteração anterior ao "congelamento" do NCO, apesar da variável com o valor da frequência ser actualizada com o valor correcto, por um erro de código este valor não estava a ser configurado no NCO. Desta forma, o NCO era "congelado" a uma frequência correspondente a esta iteração. Uma vez que a variância de frequência é elevada em situações de CNR reduzida, umas vezes este valor estaria próximo e outras vezes estaria distante da frequência do sinal de entrada, Assim se explica a volubilidade do sucesso no "congelamento" do NCO, sendo que quando a frequência do NCO era muito distante da frequência do sinal de entrada, as componentes I e Q eram cortadas pelos filtros FIR da cadeia secundária de filtragem e decimação. Se estas diferenças de frequência ultrapassassem a largura de banda destes filtros, o AGC não seria capaz de responder e consequentemente a malha não readquiriria o sincronismo.

Depois de corrigidos todos os problemas referidos anteriormente, repetiram-se os testes anteriores mas agora guardando também a diferença entre a frequência do NCO e a frequência do sinal de entrada. Os resultados obtidos, que estão apresentados na secção 7.1, estavam finalmente de acordo com os objectivos propostos para o sistema.

6. Protótipo de um Receptor Digital de Dois Canais

Como se referiu no capítulo 1, a qualidade da propagação de sinais EHF (*Extremely High Frequency*) é fortemente degradada por fenómenos meteorológicos causando atenuação, cintilação e despolarização. Este último fenómeno é consequência da assimetria das gotas de água e gelo (chuva ou nuvens de gelo) e caracteriza-se pela transferência de sinal da polarização original ou copolar (Co), para a polarização ortogonal ou crosspolar (Cx).

Conhecendo a amplitude e fase relativa do sinal crosspolar em relação ao copolar, é possível contribuir contribuir para um melhor entendimento da estrutura do canal de propagação. A caracterização estática e dinâmica do canal de propagação é portanto fundamental na implementação de sistemas de contra medida que têm como objectivo a melhoria da qualidade de serviço.

Na Figura 6.1 apresenta-se a arquitectura de um sistema orientado para a medição de dois canais, copolar e crosspolar, que combina as funções do NCO e da detecção síncrona numa única placa, mantendo a coerência entre estes dois canais.

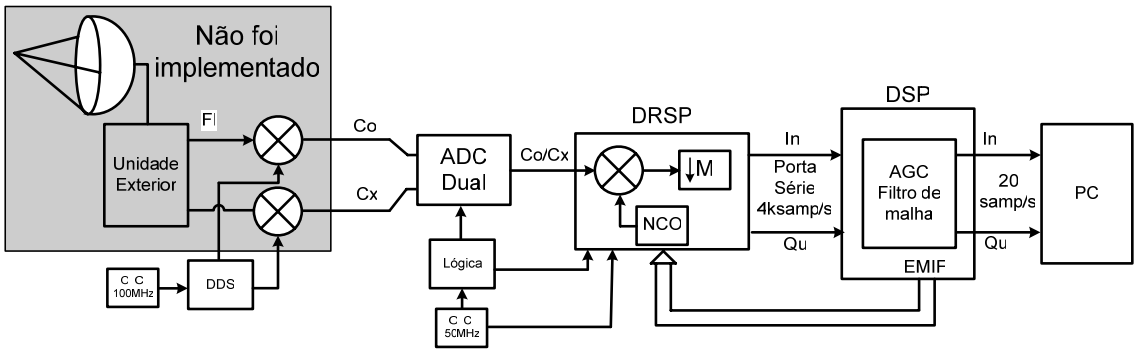

Figura 6.1 – Arquitectura do receptor digital para medição de dois canais.

Para solucionar a problemática da derivação de osciladores locais foi ainda adicionado ao sistema um módulo de síntese directa de frequência com um *chip* DDS. Este poderá então servir de oscilador local para facilitar a interface com sistemas que possuem diferentes FT's. Foi ainda adicionado um pequeno transceiver para permitir a comunicação série com o PC anfitrião.

Na secção 6.1 é feita uma descrição sobre o *hardware* para o receptor digital de dois canais. Este protótipo foi desenvolvido no âmbito de uma dissertação de mestrado [13], sendo que a minha participação se resumiu ao projecto preliminar da placa e ao desenvolvimento de código para testes de depuração de *hardware* e de *software*.

A migração do *software* anteriormente desenvolvido para o novo protótipo de dois canais ficou a meu cargo e consistiu num dos objectivos propostos para este trabalho. Este assunto foi abordado na secção 6.2, onde estão descritas todas as alterações

efectuadas no *software* que se implementou no protótipo de um canal e que está descrito em [3, cap. 4] e também no capítulo 4.

6.1. *Hardware*

Para a construção de um novo protótipo que fosse capaz de receber e processar dois sinais em simultâneo, foi necessário efectuar algumas alterações em relação ao *hardware* do protótipo de um canal que estão descritas em [3, cap. 3].

No entanto, face às excelentes prestações obtidas no passado, optou-se por manter o mesmo *chip* AD6620, dada a sua flexibilidade no processamento de dois sinais em simultâneo. Assim, a introdução de um novo sinal na cadeia de processamento não teria grande impacto nos resultados pretendidos. Por outro lado, existe ainda uma grande vantagem na utilização de um único *chip* DRSP, já que ao partilhar o mesmo NCO seria possível assegurar à partida a coerência de fase. Além disso, não seria necessária qualquer alteração na forma como a DRSP comunica com a DSP, uma vez que a programação necessária será praticamente a mesma, quer para o envio dos dados processados, como para o acesso à memória interna do *chip*.

Na implementação do novo protótipo, foram tidas em consideração as características específicas dos novos componentes, a optimização do espaço ocupado e consumo de potência, etc.

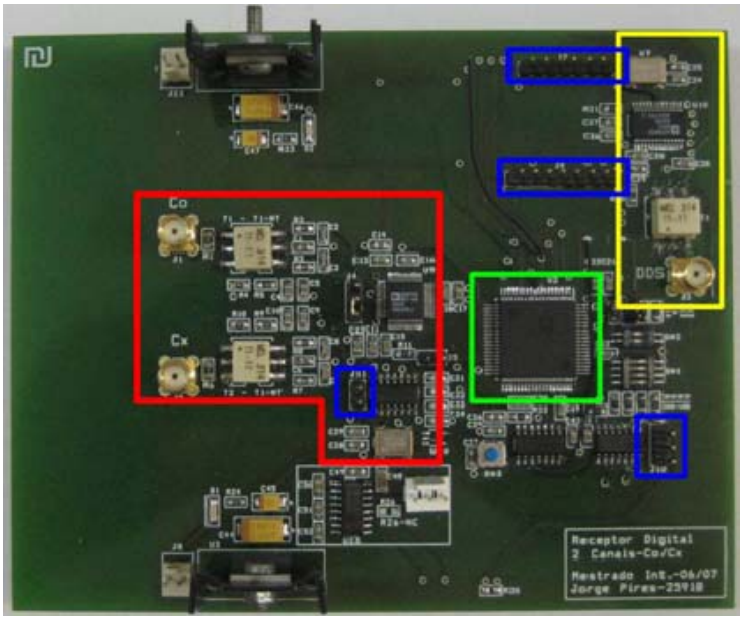

Figura 6.2 – Vista superior da placa do receptor de dois canais.

Na Figura 6.2 está representada a vista superior do *hardware* desenvolvido, onde é possível observar as entradas analógicas dos canais copolar e crosspolar e a ADC

AD9238 (a vermelho), a DRSP AD6620 (a verde), a DDS AD9850 e respectiva saída analógica (a amarelo) e os pinos de sondagem para efeitos de deouração (a azul).

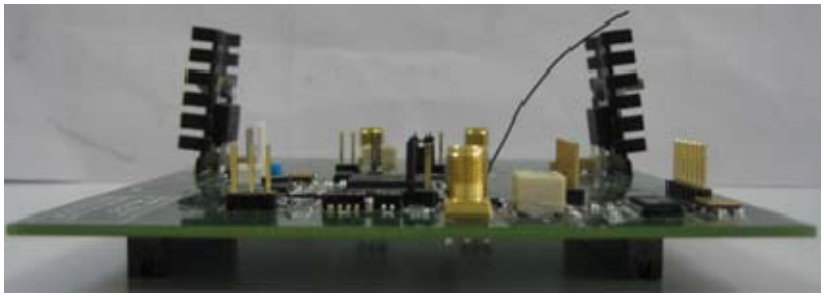

Figura 6.3 – Vista lateral da placa do receptor de dois canais.

6.1.1. Sub-Sistema ADC-DRSP

A utilização de uma ADC dual, ou seja, de *chips* com duas ADC's independentes integradas no mesmo encapsulamento, permite como é óbvio, a optimização do espaço ocupado na placa e a redução da complexidade do sistema. A utilização deste tipo de ADC's foi assumida como um progresso, na medida em que todas as ADC's estudadas em [13] apresentam um melhor desempenho face à ADC utilizada no protótipo de um canal (AD6640). No entanto, quase nenhuma possuía um módulo integrado para a multiplexagem dos sinais digitalizados, o que seria um inconveniente já que exigiria a utilização de multiplexadores adicionais.

Depois de um estudo em que foi reunida e comparada toda a informação relativa às especificações ao nível da exactidão DC e do desempenho dinâmico de algumas ADCs existentes no mercado, optou-se por utilizar a ADC dual AD9238 da *Analog Devices*, que se veio a revelar como uma boa escolha. Esta ADC tem uma resolução de 12 bits, funciona a 3.3V e a gama de tensão dos sinais analógicos à entrada é de 1Vpp a 2Vpp. Optou-se por usar uma versão de 40MSPS, já que o relógio a utilizar neste *chip* não seria superior a 25MHz.

Por outro lado, a AD9238 permite ainda a multiplexagem dos dados em qualquer uma das suas duas saídas digitais, pelo que só foi necessário utilizar uma das saídas (a outra saída foi desligada, permitindo reduzir a potência consumida e o ruído gerado). Desta forma optimizou-se o espaço ocupado e reduziu-se a complexidade do circuito na interface entre a ADC e a DRSP.

Como se pode ver Figura 6.4, a solução escolhida para gerir a multiplexagem é um processo simples e recorre a um único sinal de relógio para gerar os diversos sinais de controlo necessários na ADC (CLK_A, CLK,B e MUX_SELECT) e na DRSP (A/B), de forma a garantir o sincronismo em todo o sistema. Para que a DRSP funcione com dois canais, é necessário que estes sinais correspondam a metade do tempo de ciclo do relógio principal, pelo que se recorreu a um *flip-flop* tipo D (*chip* SN74AC74 da *Texas Instruments*) para implementar esta divisão. Foi ainda necessário inverter o sinal de

relógio enviado para o AD6620, de modo a tornar possível a aquisição das amostras no flanco ascendente.

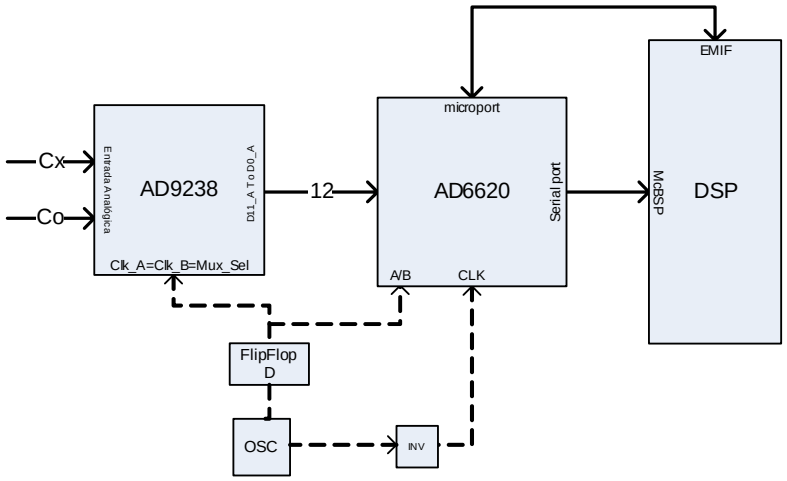

Figura 6.4 – Interface do subsistema ADS-DRSP com a ADC AD9328.

De forma a evitar os problemas ocorridos no passado na comunicação série entre a DRSP e a DSP, que estariam relacionados com a adaptação das linhas desta interface, optou-se incluir um simples *buffer* (*chip* SN74LVTH125 da *Texas Instruments*), que afectaria todas estas linhas.

6.1.2. Sub-Sistema DDS-DSP

Para efectuar a síntese digital de frequência recorreu-se à DDS AD9850 da *Analog Devices*. Este *chip* é uma DDS completa, constituída por um sistema DDS mais uma DAC, permitindo recriar um sinal sinusoidal com um valor de frequência muito preciso e com um reduzido ruído de fase.

Esta DDS é alimentada a 3.3V e permite produzir na sua saída analógica uma onda sinusoidal ou um sinal de relógio com uma determinada frequência, que pode ser ajustada com uma resolução de 32 bits (*tunning word*) até um máximo de 125MHz. A programação deste *chip* pode ser feita mediante comunicação série ou paralela.

Optou-se por programar a DDS por comunicação paralela através da interface EMIF (*External Memory Interface*) da DSP. Esta programação consiste no carregamento de 40 bits divididos em 5 palavras de 8 bits, dos quais 32 bits correspondem à *tunning word*, 5 bits correspondem à modulação de fase e os restantes 3 bits correspondem ao controlo do modo *power down*.

No entanto, como o barramento de dados da EMIF já era utilizado para a programação do *chip* AD6620 (através do espaço de endereçamento relativo ao *chip enable* CE2), houve a necessidade de partilhar este barramento com a DDS. Utilizou-se então o espaço de endereçamento reservado ao *chip enable* CE3 para endereçar a DDS

independentemente. Desta forma evitou-se a utilização de lógica de descodificação adicional.

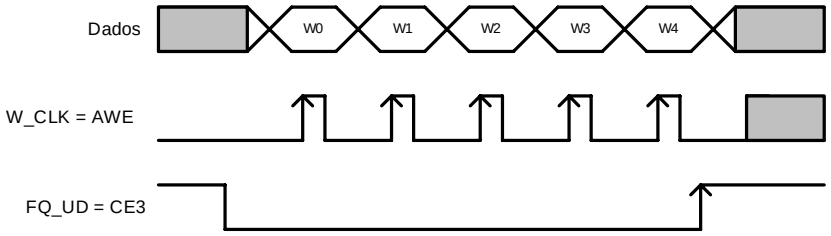

Figura 6.5 – Diagrama temporal da programação da DDS.

Como se pode ver no diagrama temporal da Figura 6.5, a escrita das palavras na memória interna do *chip* é controlada pelos sinais W_CLK e FQ_UD. A transição descendente do sinal FQ_UD sinaliza o início do carregamento dos 40 bits e inicializa o ponteiro de endereço para o primeiro registo interno do *chip*. Posteriormente, os flancos ascendentes do sinal W_CLK vão sinalizar o carregamento de cada uma das 5 palavras de 8 bits, movendo o ponteiro para o registo seguinte. Após terem sido enviadas as 5 palavras, os flancos do sinal W_CLK são ignorados até ser executado um *reset* ou até surgir um novo flanco descendente do FQ_UD.

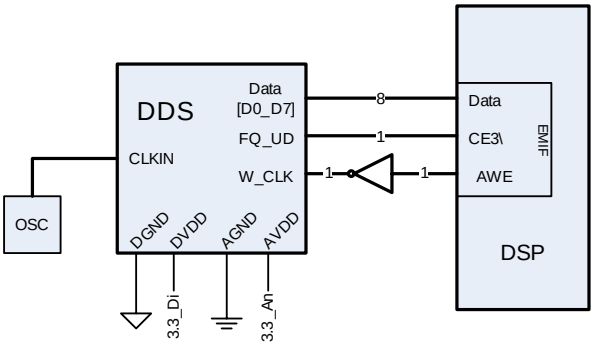

Figura 6.6 – Interface do subsistema DDS-DRSP com a DDS AD9850.

Na Figura 6.6 estão representadas as ligações entre a DDS e a DSP, onde se pode observar que os sinais CE3\ e AWE provenientes da DSP, ligam respectivamente aos sinais FQ_UD e W_CLK da DDS. No entanto, houve ainda a necessidade de inverter o sinal AWE uma vez que este é *active low* ao contrário do sinal W_CLK que é *active high*.

6.1.3. Resumo dos Resultados de Avaliação da Placa

No âmbito da tese referida anteriormente os resultados experimentais do protótipo foram os seguintes:

- Densidade espectral de ruído de fundo relativo ao sinal máximo de entrada: +130dBc (portanto um valor residual tendo em atenção o valor intrinseco do sinal da baliza);
- Isolamento entre canais: superior a 90dB;
- Linearidade de amplitude em malha aberta: melhor que 0.2dB em 30dB de gama e melhor que 0.5dB em 60dB de gama;
- Diferença de fase relativa: melhor que 1° em 60dB de gama.

6.2. Migração de *Software* para o Protótipo de Dois Canais

As alterações de *software* mais importantes são consequência de se utilizar neste protótipo um oscilador a cristal de 50MHz, em vez de 40MHz, e do AD6620 ter que processar agora dois canais, em vez de um único canal. No entanto, a adição do *chip* DDS ao sistema, bem como do *hardware* para permitir o envio de dados em tempo real pela porta série, exigiram também algumas mudanças no *software* desenvolvido.

6.2.1. Alterações na Configuração da EMIF

Para configurar a interface da EMIF com os parâmetros de configuração desejados, é utilizado o módulo em linguagem GEL – "*dsk6xinit.gel*"–, que é executado durante o *startup* do *Code Composer Studio*.

A alteração na configuração desta interface deve-se à necessidade de programação da DDS. O registo de controlo do *chip enable* CE3 da EMIF deve ser programado correctamente de modo a que os sinais AWE e CE3 verifiquem os requisitos temporais e possam realizar as operações pretendidas tal como foram descritas na secção 6.1.2.

Para que a programação da DDS seja executada correctamente, de acordo com os valores mínimos para os ciclos de escrita do sinal W_CLK referidos em [14], cada ciclo de escrita no espaço de endereçamento reservado ao CE3 tem que respeitar a seguinte condição:

$$Write\ Strobe \geq 7ns\,. \tag{6.1}$$

Como o sinal de relógio da EMIF (ECLKOUT) apresenta uma frequência de 100MHz, a que corresponde um período de 10ns, basta garantir que o valor correspondente ao ciclo de escrita seja superior a um ciclo da EMIF.

Assim, de acordo com o que está descrito em [15, pág. 89], configurou-se o registo CECTL3 de controlo do CE3 com o valor 0x44F10221, ao qual correspondem os seguintes parâmetros:

- MTYPE = 0010b – Memória assíncrona de 32 bits;
- WRSETUP = 0100b – *Write Setup* de 4 ciclos de ECLKOUT;
- WRSTRB = 010011b – *Write Strobe* de 19 ciclos de ECLKOUT;

- WRHOLD = 11b – *Write Hold* de 3 ciclos de ECLKOUT;
- RDSETUP = 0001b – *Read Setup* de 1 ciclo de ECLKOUT (indiferente);
- RDSTRB = 000010b – *Read Strobe* de 2 ciclos de ECLKOUT(indiferente);
- RDHOLD = 0001b – *Read Hold* de 1 ciclo de ECLKOUT (indiferente);
- TA = 00b – Tempo de *Turn-Around* nulo.

6.2.2. Alterações na Configuração do Porto McBSP1

Ao contrário do que acontecia com o protótipo de um canal, o porto McBSP1 não vai ser usado apenas para recepção de dados provenientes do AD6620, mas também para o envio de dados em tempo real para o PC anfitreão. Assim, além de alterar o valor do registo o registo de controlo de recepção, RCR, foi também necessário alterar o valor do registo de controlo de transmissão, XCR.

No protótipo de dois canais, as amostras provenientes do AD6620 são agora constituidas por 64 bits, em que os dados do canal A correspondem aos primeiros 32 bits, sendo que os restantes correspondem aos dados do canal B. Cada um destes grupos de 32 bits está obviamente dividido em 16 bits de dados da componente I e em 16 bits de dados da componente Q.

Optou-se então por fazer a recepção das amostras de 64 bits em duas fases, em que em cada fase seriam recebidos 32 bits. Os dados I e Q em cada fase seriam separados posteriormente no próprio programa.

Assim, de acordo com o que está descrito em [16, págs. 86-87], o registo RCR foi configurado com o valor 0x000101A0, ao qual correspondem os seguintes parâmetros:

- RPHASE = 0b – existe apenas uma fase de recepção (*single-phase frame*), a fase 1;
- RFRLEN2 = 0000000b – valor por defeito (indiferente);
- RWDLEN2 = 000b – valor por defeito (indiferente);
- RCOMPAND = 00b – não há *companding* e a transferência dos bits é iniciada pelo bit mais significativo (*MSB first*);
- RFIG = 0b – a recepção de um impulso de sincronização de frame inesperado após o início de uma transmissão não é ignorado, ou seja, é iniciada uma nova transferência logo a seguir a este impulso;
- RDATDLY = 01b – atraso de 1 bit na recepção dos dados, que corresponde ao intervalo de tempo entre a transição *low-high* do impulso FSR e o início da transmissão do primeiro bit da amostra. No AD6620, este atraso corresponde a um ciclo de relógio, ou seja, ao tempo de transmissão de 1 bit;
- RFRLEN1 = 00000001b – a recepção de dados é de dois elemento na fase 1;
- RWDLEN1 = 101b – o tamanho dos elementos na fase 1 é de 32 bits;

- RWDREVRS = 0b – a recepção dos dados é feita pela ordem convencional (*MSB first*), de acordo com a ordem de saída dos bits no *chip* AD6620. O modo de recepção reversa (*LSB first*) está desactivado.

Para a transmissão de dados pelo porto McBSP1, os parâmetros do registo XCR, que estão descritos em [16, págs. 88-89], devem ser configurados de acordo com a solução encontrada e referida em [8]. No entanto, por falta de tempo, não foi possível testar esta solução, pelo que esta é uma questão a ser discutida no futuro.

6.2.3. Alterações na Configuração do AD6620

Com a passagem a um protótipo de dois canais, a principal alteração na configuração do *chip* AD6620 está no modo de funcionamente, que passou de *Single Channel Real Mode* para *Dual Channel Real Mode*. Para tal tiveram que se alterar os três valores de configuração do registo MODE CONTROL REGISTER (ver secção 6.2.5).

Relativamente ao projecto do filtro RCF, a alteração da frequência do sinal de relógio para o *chip* AD6620 também deve ser considerada, uma vez que as frequências de amostragem à entrada e à saída do filtro também se alteram. Assim, é necessário calcular novos coeficientes para que o filtro possa cumprir as especificações desejadas.

De acordo com [17], a frequência de amostragem à entrada deste filtro é dada por:

$$f_{SAMP5} = \frac{f_{SAMP}}{M_{CIC2} M_{CIC5}} = \frac{25 \times 10^6}{8 \times 32} = 97.656\text{KHz}\,. \qquad (6.2)$$

e a frequência de amostragem à saída é dada por:

$$f_{SAMPR} = \frac{f_{SAMP}}{M_{CIC2} M_{CIC5} M_{RCF}} = \frac{25 \times 10^6}{8 \times 32 \times 32} = 3052\text{Hz}\,, \qquad (6.3)$$

onde f_{SAMP} corresponde a metade da frequência de relógio do AD6620 (no caso da utilização de dois canais) e M_{CIC2}, M_{CIC5} e M_{RCF} representam respectivamente os factores de decimação dos filtros CIC2, CIC5 e RCF.

Assim, o filtro RCF é um filtro FIR passa-baixo que deve ser projectado com uma frequência de amostragem que corresponda à frequência à entrada do filtro (97.656kHz) e com uma frequência de atenuação que corresponda a metade da frequência de amostragem à saída (1526Hz). Definiu-se uma frequência de corte de aproximadamente 600Hz e o ganho do filtro na banda de atenuação deveria estar entre -25 e -30dBs.

De salientar ainda que o único registo de configuração do AD6620 que tem que ser alterado é o registo que define o número de TAPS. Esta alteração é necessária uma vez que o número máximo de coeficientes a usar para o filtro RCF diminui para metade quando são processados dois canais, pelo que o número de TAPS passa agora a ser limitado a 128.

Projectou-se então este filtro tal como vem referido em [3, cap. 4].

No gráfico da Figura 6.7 está representada a resposta em frequência do filtro RCF projectado e cujos coeficientes se podem observar na Tabela 6.1.

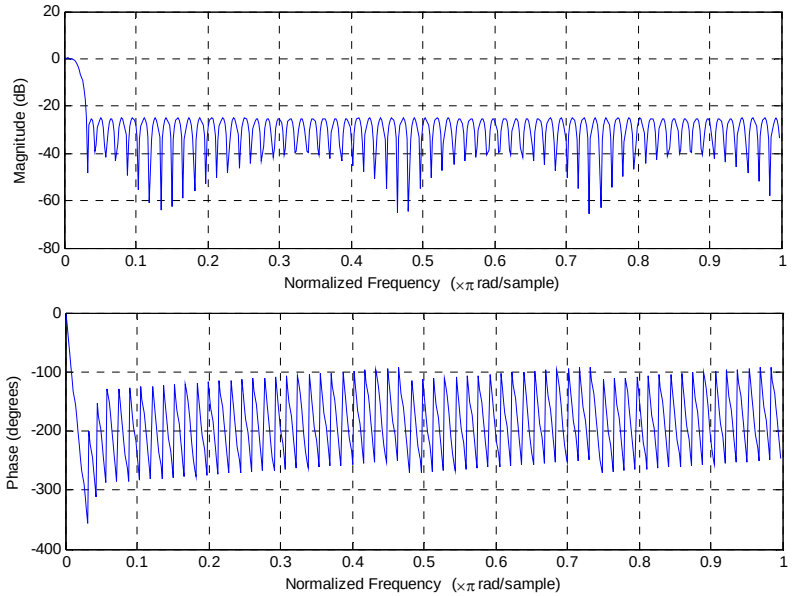

Figura 6.7 – Resposta em frequência do filtro RCF do AD6620.

n	$h_1(n)$	*n*	$h_1(n)$	*n*	$h_1(n)$	*n*	$h_1(n)$
0	-15056	**32**	3731	**64**	11942	**96**	3388
1	-1186	**33**	4086	**65**	11919	**97**	3041
2	-1219	**34**	4446	**66**	11875	**98**	2716
3	-1242	**35**	4810	**67**	11804	**99**	2397
4	-1257	**36**	5178	**68**	11713	**100**	2085
5	-1261	**37**	5547	**69**	11600	**101**	1784
6	-1255	**38**	5917	**70**	11464	**102**	1496
7	-1238	**39**	6286	**71**	11307	**103**	1223
8	-1208	**40**	6654	**72**	11130	**104**	963
9	-1164	**41**	7020	**73**	10934	**105**	717
10	-1105	**42**	7381	**74**	10718	**106**	484
11	-1031	**43**	7738	**75**	10485	**107**	266
12	-941	**44**	8086	**76**	10234	**108**	62
13	-837	**45**	8425	**77**	9966	**109**	-128
14	-726	**46**	8759	**78**	9684	**110**	-302
15	-623	**47**	9078	**79**	9387	**111**	-453
16	-453	**48**	9387	**80**	9078	**112**	-623
17	-302	**49**	9684	**81**	8759	**113**	-726
18	-128	**50**	9966	**82**	8425	**114**	-837
19	62	**51**	10234	**83**	8086	**115**	-941
20	266	**52**	10485	**84**	7738	**116**	-1031
21	484	**53**	10718	**85**	7381	**117**	-1105
22	717	**54**	10934	**86**	7020	**118**	-1164
23	963	**55**	11130	**87**	6654	**119**	-1208

24	1223	**56**	11307	**88**	6286	**120**	-1238
25	1496	**57**	11464	**89**	5917	**121**	-1255
26	1784	**58**	11600	**90**	5547	**122**	-1261
27	2085	**59**	11713	**91**	5178	**123**	-1257
28	2397	**60**	11804	**92**	4810	**124**	-1242
29	2716	**61**	11875	**93**	4446	**125**	-1219
30	3041	**62**	11919	**94**	4086	**126**	-1186
31	3388	**63**	11942	**95**	3731	**127**	-15056

Tabela 6.1 – Coeficientes do filtro RCF do AD6620.

6.2.4. Alterações na Cadeia Secundária de Filtragem e Decimação

Com uma nova frequência de amostragem à saída do detector de fase, tornou-se necessário projectar novos coeficientes para os filtros FIR da cadeia secundária de filtragem e decimação, para que estes continuassem a cumprir os requisitos propostos na secção 3.2. Optou-se por definir a largura de banda do AGC num valor próximo dos 20Hz, pelo que foi necessário reduzir o factor de decimação total na cadeia secundária de filtragem e decimação para M_{TOTAL}=150. Para tal, reduziu-se o factor de decimação do filtro FIR1 para M_1=30 e manteve-se o factor de decimação do filtro FIR2 (M_2=5).

Assim, analogamante ao que foi descrito na secção 3.2, o filtro FIR1 foi projectado com uma frequência de corte de 25Hz e com uma frequência de atenuação de 85Hz, enquanto que o filtro FIR2 foi projectado com uma frequência de corte de 10.17Hz e uma frequência de atenuação de 16Hz.

Na Figura 6.8 e na Figura 6.9 estão representados os gráficos das respostas em frequência dos filtros FIR projectados e cujos coeficientes se apresentam na Tabela 6.2 e na Tabela 6.3.

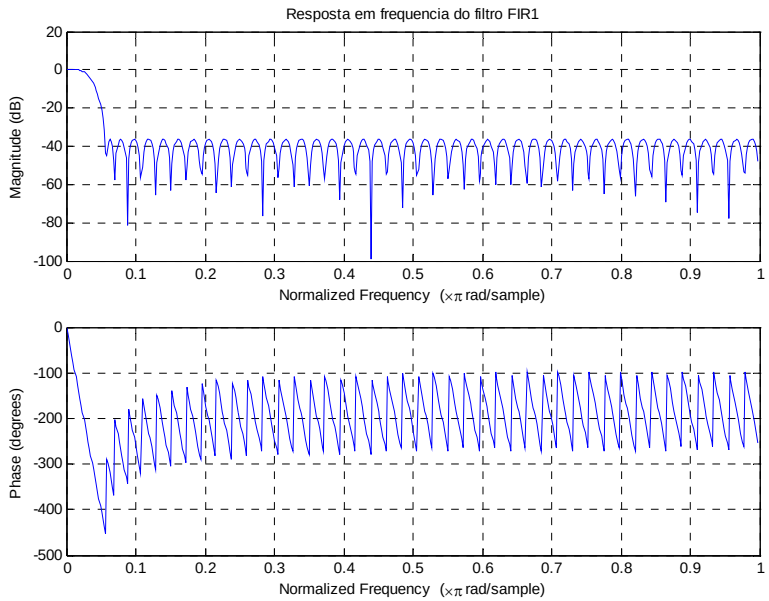

Figura 6.8 – Resposta em frequência do filtro FIR1 para o receptor de dois canais.

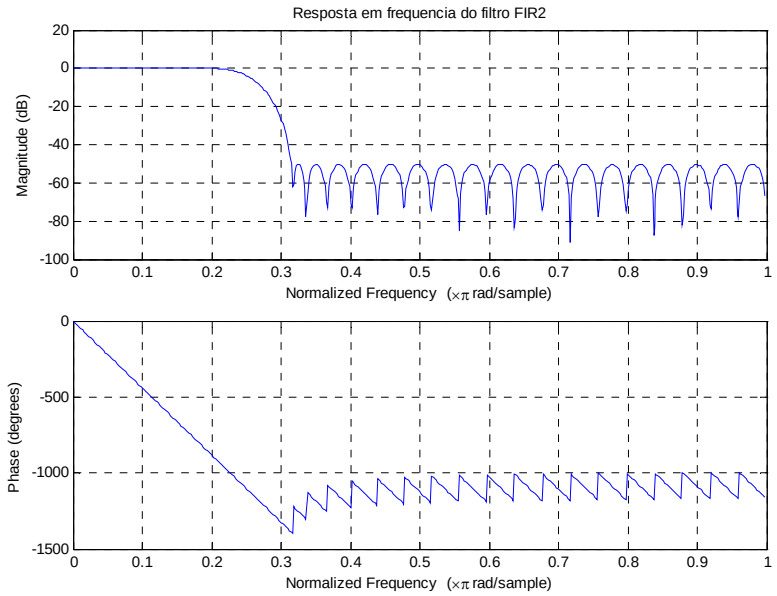

Figura 6.9 – Resposta em frequência do filtro FIR2 para o receptor de dois canais.

n	$h_1(n)$	*n*	$h_1(n)$	*n*	$h_1(n)$	*n*	$h_1(n)$
0	-0.00869309	23	0.00722028	46	0.03692186	69	0.00288082
1	-0.00251093	24	0.00886534	47	0.03652472	70	0.00165629
2	-0.00282210	25	0.01058644	48	0.03593536	71	0.00055472
3	-0.00311501	26	0.01238055	49	0.03515874	72	-0.00042618
4	-0.00338369	27	0.01422667	50	0.03420595	73	-0.00127699
5	-0.00361407	28	0.01610876	51	0.03308442	74	-0.00201069
6	-0.00380238	29	0.01801408	52	0.03181196	75	-0.00261024
7	-0.00393704	30	0.01992045	53	0.03039772	76	-0.00311209
8	-0.00401324	31	0.02181439	54	0.02886021	77	-0.00348454
9	-0.00401332	32	0.02367312	55	0.02721359	78	-0.00374990
10	-0.00392949	33	0.02547971	56	0.02547971	79	-0.00392949
11	-0.00374990	34	0.02721359	57	0.02367312	80	-0.00401332
12	-0.00348454	35	0.02886021	58	0.02181439	81	-0.00401324
13	-0.00311209	36	0.03039772	59	0.01992045	82	-0.00393704
14	-0.00261024	37	0.03181196	60	0.01801408	83	-0.00380238
15	-0.00201069	38	0.03308442	61	0.01610876	84	-0.00361407
16	-0.00127699	39	0.03420595	62	0.01422667	85	-0.00338369
17	-0.00042618	40	0.03515874	63	0.01238055	86	-0.00311501
18	0.00055472	41	0.03593536	64	0.01058644	87	-0.00282210
19	0.00165629	42	0.03652472	65	0.00886534	88	-0.00251093
20	0.00288082	43	0.03692186	66	0.00722028	89	-0.00869309
21	0.00421797	44	0.03712214	67	0.00566872		
22	0.00566872	45	0.03712214	68	0.00421797		

Tabela 6.2 – Coeficientes do primeiro filtro FIR para o receptor de dois canais.

n	$h_2(n)$	*n*	$h_2(n)$	*n*	$h_2(n)$	*n*	$h_2(n)$
0	0.00202862	**13**	0.00063760	**26**	0.19854921	**39**	-0.00757529
1	0.00047787	**14**	0.01637971	**27**	0.11167886	**40**	0.00089251
2	-0.00110600	**15**	0.02545611	**28**	0.02476208	**41**	0.00637729
3	-0.00292611	**16**	0.01838796	**29**	-0.03377642	**42**	0.00660671
4	-0.00333950	**17**	-0.00582690	**30**	-0.05170794	**43**	0.00300696
5	-0.00117082	**18**	-0.03593549	**31**	-0.03593549	**44**	-0.00117082
6	0.00300696	**19**	-0.05170794	**32**	-0.00582690	**45**	-0.00333950
7	0.00660671	**20**	-0.03377642	**33**	0.01838796	**46**	-0.00292611
8	0.00637729	**21**	0.02476208	**34**	0.02545611	**47**	-0.00110600
9	0.00089251	**22**	0.11167886	**35**	0.01637971	**48**	0.00047787
10	-0.00757529	**23**	0.19854921	**36**	0.00063760	**49**	0.00202862
11	-0.01346855	**24**	0.25275566	**37**	-0.01116412		
12	-0.01116412	**25**	0.25275566	**38**	-0.01346855		

Tabela 6.3 – Coeficientes do segundo filtro FIR para o receptor de dois canais.

6.2.5. Outras Alterações

Nas secções anteriores foram descritas as alterações ao *software* devido ao processamento de dois canais e à adição do *chip* DDS e do *hardware* para o envio de dados em tempo real. Estas alterações obrigaram à criação ou alteração das seguintes variáveis e constantes:

Constantes:

- *AD9850_ADDR* = 0xB0000000 – endereço base para o banco de memória CE3 reservado para a DDS (*chip* AD9850);
- *FREQ_CLK* = 25000000 – metade do valor da frequência do relógio do *chip* AD6620 (50MHz);
- MODE_CTRL_VAL1 = 0x01 – valor para o primeiro passo de configuração do registo interno MODE CONTROL REGISTER, que coloca o *chip* AD6620 em *Soft Reset*;
- MODE_CTRL_VAL2 = 0x03 – valor para o segundo passo de configuração do registo interno MODE CONTROL REGISTER, que configura o modo de entrada como *Dual Channel Real Mode*, configura o *chip* AD6620 como *Sync Slave* e o mantém em *Soft Reset*;
- MODE_CTRL_VAL3 = 0x02 – valor para o terceiro passo de configuração do registo interno MODE CONTROL REGISTER, que mantém os valores configurados no passo anterior mas que coloca o *chip* AD6620 fora do *Soft Reset*, ou seja, em funcionamento;
- *N_TAPS* = 127 – valor para o registo interno NTAPS-1 do *chip* AD6620;
- *N_COEFS* = 128 – número de coeficientes para o filtro RCF do *chip* AD6620;

▫ *FC_RCF* = 600 – valor da frequência de corte do filtro RCF do *chip* AD6620;

▫ *M1* = 30 – factor de decimação do filtro FIR1;

▫ *M2* = 5 – factor de decimação do filtro FIR2;

▫ *MTOTAL*.= 150 – factor de decimação total dos filtros FIR1 e FIR2;

▫ *N_AMOSTRAS_FIR1* = 90 – nº de elementos do *buffer* circular do filtro FIR1;

▫ *N_AMOSTRAS_FIR2* = 50 – nº de elementos do *buffer* circular do filtro FIR2;

▫ *N_AMOSTRAS_INICIAIS* = 1410 – nº de amostras necessárias para o preenchimento inicial dos *buffers* circulares dos filtros FIR.

- Variáveis globais:

  ▫ *short amostraI_Co* – variável para armazenar a amostra I do sinal copolar;

  ▫ *short amostraQ_Co* – variável para armazenar a amostra Q do sinal copolar;

  ▫ *short amostraI_Cx* – variável para armazenar a amostra I do sinal crosspolar;

  ▫ *short amostraQ_Cx* – variável para armazenar a amostra Q do sinal crosspolar;

  ▫ *float amostraI_Co_out* – amostra I do sinal copolar na saída da cadeia de filtragem e decimação;

  ▫ *float amostraQ_Co_out* – amostra Q do sinal copolar na saída da cadeia de filtragem e decimação;

  ▫ *float amostraI_Cx_out* – amostra I do sinal crosspolar na saída da cadeia de filtragem e decimação;

  ▫ *float amostraQ_Cx_out* – amostra Q do sinal crosspolar na saída da cadeia de filtragem e decimação;

  ▫ *BUFFER *structFIR1_I_Co*, **structFIR1_Q_Co* – ponteiros para estruturas de *buffers* circulares para as amostras I e Q do sinal copolar no filtro FIR1;

  ▫ *BUFFER *structFIR1_I_Cx*, **structFIR1_Q_Cx* – ponteiros para estruturas de *buffers* circulares para as amostras I e Q do sinal crosspolar no filtro FIR1;

  ▫ *BUFFER *structFIR2_I_Co*, **structFIR2_Q_Co* – ponteiros para estruturas de *buffers* circulares para as amostras I e Q do sinal copolar no filtro FIR2;

  ▫ *BUFFER *structFIR2_I_Cx*, **structFIR2_Q_Cx* – ponteiros para estruturas de *buffers* circulares para as amostras I e Q do sinal crosspolar no filtro FIR2;

  ▫ *float bufferFIR1_I_Co[N_AMOSTRAS_FIR1]* – *array* para armazenar as amostras I do sinal copolar à entrada do filtro FIR1;

  ▫ *float bufferFIR1_Q_Co[N_AMOSTRAS_FIR1]* – *array* para armazenar as amostras Q do sinal copolar à entrada do filtro FIR1;

▫ *float bufferFIR1_I_Cx[N_AMOSTRAS_FIR1]* – *array* para armazenar as amostras I do sinal crosspolar à entrada do filtro FIR1;

▫ *float bufferFIR1_Q_Cx[N_AMOSTRAS_FIR1]* – *array* para armazenar as amostras Q do sinal crosspolar à entrada do filtro FIR1;

▫ *float bufferFIR2_I_Co[N_AMOSTRAS_FIR2]* – *array* para armazenar as amostras I do sinal copolar à entrada do filtro FIR2;

▫ *float bufferFIR2_Q_Co[N_AMOSTRAS_FIR2]* – *array* para armazenar as amostras Q do sinal copolar à entrada do filtro FIR2;

▫ *float bufferFIR2_I_Cx[N_AMOSTRAS_FIR2]* – *array* para armazenar as amostras I do sinal crosspolar à entrada do filtro FIR2;

▫ *float bufferFIR2_Q_Cx[N_AMOSTRAS_FIR2]* – *array* para armazenar as amostras Q do sinal crosspolar à entrada do filtro FIR2;

▫ *double amostraI_Co_AGC* – amostra I do sinal copolar na saída do AGC;

▫ *double amostraQ_Co_AGC* – amostra Q do sinal copolar na saída do AGC.

- Variáveis locais:

  ▫ *int amostraCo* – variável local da rotina *RSI_RINT1()* para armazenar a amostra de 32 bits do sinal copolar, recebida pela porta série;

  ▫ *int amostraCx* – variável local da rotina *RSI_RINT1()* para armazenar a amostra de 32 bits do sinal crosspolar, recebida pela porta série;

  ▫ *float auxI_Cx*, *auxI_Co*, *auxQ_Cx*, *auxQ_Co* – variáveis auxiliares das rotinas *filtDecimInic()* e *filtDecim()* para os cálculos dos filtros FIR relativos às componentes I e Q dos sinais copolar e crosspolar.

Foi então necessário efectuar algumas alterações de código nas seguintes rotinas:

- *RSI_RINT1()*:

A leitura das amostras de 64 bits recebidas pelo por série McBSP1 passa agora a ser executada em dois passos, em que primeiro são recebidos os primeiros 32 bits que são relativos ao sinal crosspolar e posteriormente são recebidos os últimos 32 bits relativos ao sinal copolar. Assim, a primeira leitura é efectuada para a nova variável local *amostraCx*, enquanto que a segunda é efectuada para a nova variável local *amostraCo*, que substitui a antiga variável *amostra*.

Posteriormente, a divisão destas duas amostras de 32 bits em componentes I e Q é efectuada tal como no protótipo de um canal. As amostras I e Q do sinal copolar são armazenadas nas novas variáveis *amostraI_Co* e *amostraQ_Co*, que substituiem as variáveis antigas *amostraI* e *amostraQ*. As amostras I e Q do sinal crosspolar são armaxenadas nas variáveis *amostraI_Cx* e *amostraQ_Cx*.

- *filtDecimInic()* e *filtDecim()*:

Todos os cálculos efectuados para o protótipo de um canal tiveram que ser replicados para dois canais. Desta forma, as instruções nestas rotinas foram repetidas tendo em conta as novas variáveis criadas para o processamento do sinal copolar e crosspolar.

De referir ainda que, relativamente ao *software pipelining* dos ciclos relativos aos cálculos dos filtros FIR, o compilador aconselhou a utilização da opção *–mh8*. Comparando este valor com o que se referiu na secção 5.2, facilmente se conclui que pelo facto de se terem replicado as instruções nestes ciclos, a quantidade de *data memory padding* que assegura o melhor desempenho na execução do ciclo passou a ser o dobro.

- *AGC_digital()*:

O AGC apenas actuará sobre as componentes I e Q do sinal copolar, pelo que o cálculo da saída do AGC, ou seja, da amplitude detectada, é efectuado sobre as componentes I e Q deste sinal à saída dos filtros FIR. Assim, as variáveis *amostraI_out* e *amostraQ_out* foram substituidas pelas novas variáveis *amostraI_Co_out* e *amostraQ_Co_out*. Por outro lado, as variáveis *amostraI_AGC* e *amostraQ_AGC* foram alteradas para *amostraI_Co_AGC* e *amostraQ_Co_AGC*.

- *PLL_digital()*:

Os cálculos do filtro digital da PLL são feitos sobre a componente Q do sinal copolar actuada pelo AGC, pelo que a variável antiga *amostraQ_AGC* foi alterada para *amostraQ_Co_AGC*. Assim, as variáveis *amostraI_out* e *amostraQ_out* foram substituidas pelas novas variáveis *amostraI_Co_out* e *amostraQ_Co_out*.

- *lockIndication()*:

O cálculo da running average para a indicação de sincronismo é efectuado sobre a componente I do sinal copolar actuada pelo AGC, pelo que a variável antiga *amostraI_AGC* foi alterada para *amostraI_Co_AGC*.

- *CNR_estimation()*:

O cálculo da *running average* para estimação de CNR utiliza as componentes I e Q do sinal copolar à saída dos filtros FIR, pelo que, tal como na rotina *AGC_digital()*, as variáveis *amostraI_out* e *amostraQ_out* foram substituidas pelas novas variáveis *amostraI_Co_out* e *amostraQ_Co_out*.

- *startSystem()*:

Na inicialização dos *buffers* circulares dos filtros FIR, os *buffers* antigos foram substituídos pelos relativos ao sinal copolar e foram ainda adicionadas instruções para a inicialização dos *buffers* associados ao sinal crosspolar.

7. Resultados

Neste capítulo são apresentados os resultados mais importantes que pretendem demonstrar o funcionamento do sistema desenvolvido neste trabalho. Estes resultados foram obtidos com o sistema completamente funcional, após a conclusão dos testes referentes ao *debugging* e que estão descritos no capítulo 5.

7.1. Comportamento Dinâmico

Para avaliar o comportamento dinâmico efectuou-se um teste do sistema em funcionamento sem proceder ao "congelamento" do NCO, com um sinal de entrada adicionado de ruído e modulado com uma frequência f_{mod} de aproximadamente 0.03Hz. Utilizou-se ainda um atenuador fixo de 10dB intercalado entre o gerador de sinal e o *power splitter*.

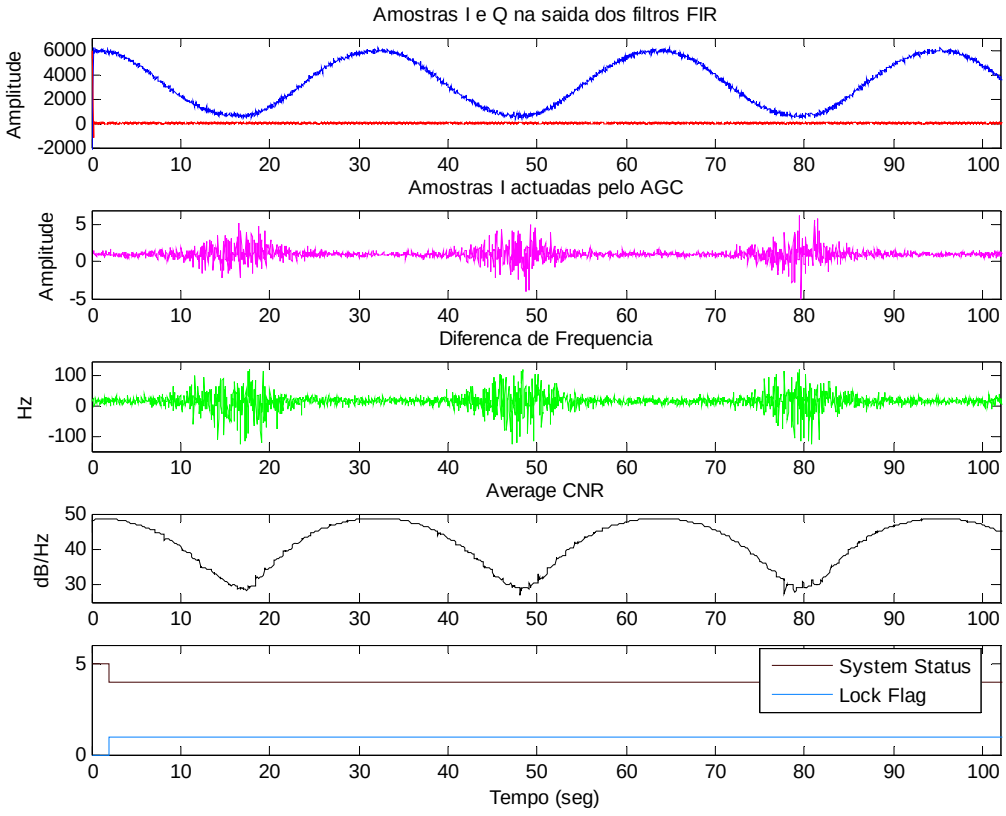

Figura 7.1 – Resultados obtidos com um atenuador de 10dB e sem "congelar" o NCO.

Como se pode observar no gráfico da Figura 7.1, apesar da amplitude do sinal de entrada atingir valores demasiado baixos, a malha manteve o sincronismo mesmo com a CNR a atingir valores próximos de 30dB/Hz. De acordo com o que se esperaria, a variância de frequência do NCO e da componente I actuada pelo AGC aumenta

consideravelmente sempre que a CNR diminui. No entanto, a malha mantém o sincronismo mesmo com a diferença de frequência a atingir picos de 128Hz. A média desta diferença é de aproximadamente 16Hz, o que corresponderia à diferença de frequência entre o NCO e oscilador do gerador de sinal. Por outro lado, a média da componente I actuada pelo AGC é de cerca de 1.004, o que indica o bom funcionamento do AGC, pelo que consequentemente a indicação de sincronismo é sempre positiva.

A atenuação total do sinal é de cerca de 30dB, o que corresponde à atenuação introduzida pela modulação somada ao valor do atenuador fixo. O sinal de entrada foi gerado de modo a apresentar um valor CNR próximo de 55dB/Hz. Assim, seria de esperar que o valor de CNR estimado durante o teste variasse entre 25 e 45dB/Hz mas no entanto, este valor oscila entre 28 e 48dB/Hz. Esta diferença de 3dB/Hz poderá ser devida a erros na medição da CNR do sinal de entrada e/ou a erros na estimação da densidade espectral de potência de ruído, efectuada no domínio dos tempos durante a estimação da frequência do sinal de entrada.

Será de esperar que a variância de frequência aumente bastante quando a atenuação do sinal é ainda maior, o que poderá levar à perda completa de sincronismo e consequentemente constituir um problema para o sistema, Estes resultados permitiram confirmar a necessidade de efectuar o "congelamento" do NCO sempre que a CNR descer abaixo de um determinado patamar e houver indicação de perda de sincronismo. Embora a PLL seja capaz de manter o sincronismo para valores de CNR muito reduzidos, este patamar deverá rondar os 27dB/Hz de modo a garantir uma certa margem de confiança.

Repetiu-se o teste anterior com f_{mod}=0.03Hz, mas agora com o sistema a proceder ao "congelamento" do NCO quando a CNR descer abaixo de 30dB/Hz e com um limite mínimo de 0.99 para a indicação de sincronismo. Os resultados obtidos estão representados na Figura 7.2.

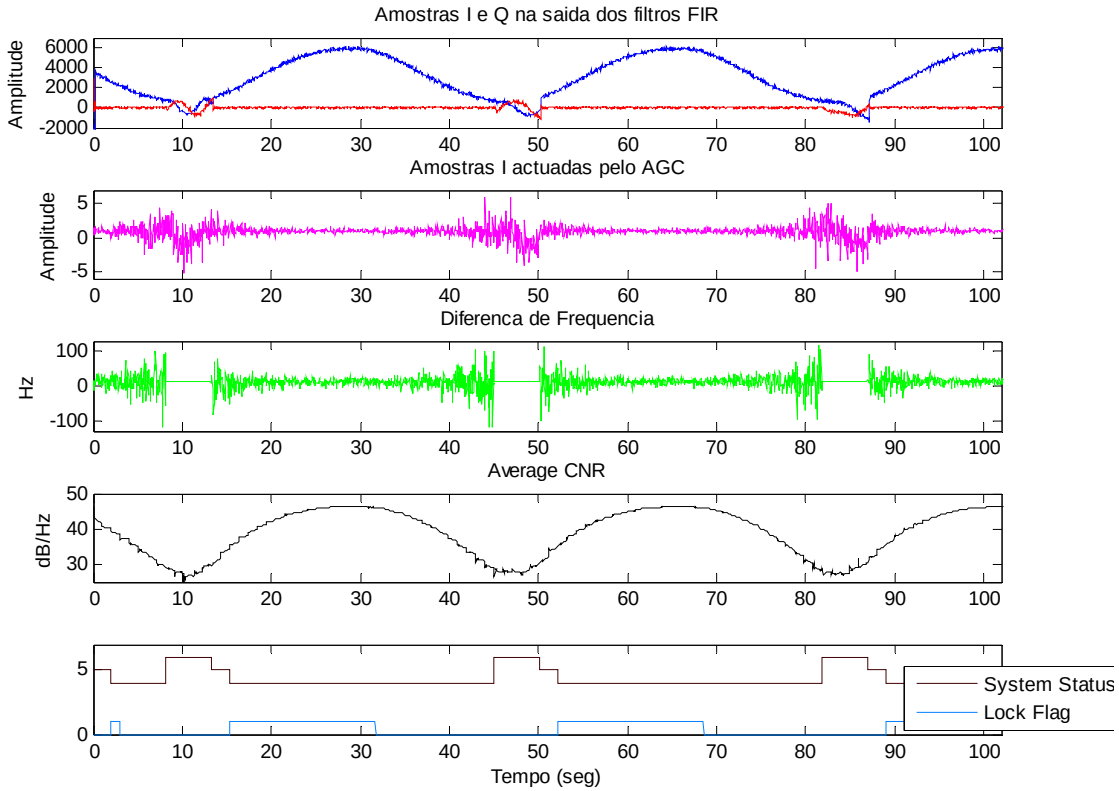

Figura 7.2 – Resultados obtidos com um atenuador de 10dB e a "congelar" o NCO (f_{mod}=0.03Hz).

Como se pode observar, os resultados são bastante satisfatórios já que o "congelamento" do NCO é sempre efectuado com sucesso, sendo que a malha readquire sempre o sincronismo. O desvio de frequência mantém-se contante durante o período em que o NCO está "congelado", uma vez que a malha é aberto e o NCO é configurado com uma frequência correspondente à média de frequência calculada a cada 5 segundos e validada de acordo com o valor médio de CNR a cada segundo. Desta forma, evita-se que o NCO sofra desvios de frequência que possam ser significativos e ao mesmo tempo, se o sinal de entrada não sofrer desvios consideráveis de frequência, a malha poderá facilmente readquirir o sincronismo.

De seguida repetiu-se mais uma vez o teste anterior, mas agora com f_{mod}=0.009Hz de modo a observar o comportamento do sistema com um período de "congelamento" do NCO mais alargado. Os resultados obtidos estão representados na Figura 7.3.

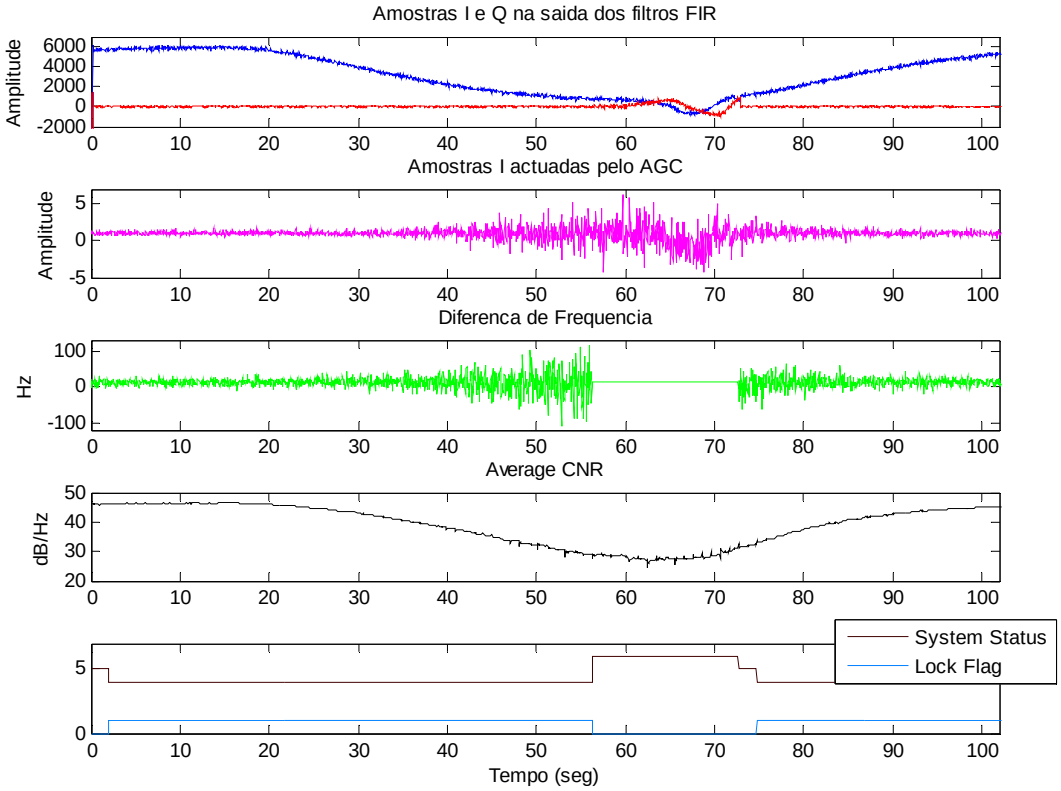

Figura 7.3 – Resultados obtidos com um atenuador de 10dB e a "congelar" o NCO (f_{mod}=0.009Hz).

Com uma variação mais lenta da atenuação do sinal, pode-se observar mais facilmente o comportamento das componentes I e Q durante o "congelamento" do NCO. Nestes testes a frequência do sinal de entrada não é alterada ao longo do tempo, pelo que a malha readquiria o sincronismo mesmo que o período de "congelamento" do NCO fosse bastante mais prolongado do que o que se observou neste teste. No entanto, numa situação real, mesmo que o sinal apresente uma variação lenta de frequência, o NCO não poderá estar "congelado" durante um período de tempo indefinido. Nestas situações, o sinal de entrada poderia sofrer desvios importantes de frequência e consequentemente, quando a CNR voltasse a subir acima do patamar mínimo, a malha poderia não conseguir readquirir o sincronismo. Assim, pode-se concluir que é muito importante limitar o tempo de espera pela recuperação do sinal, ou seja, o período de "congelamento" do NCO.

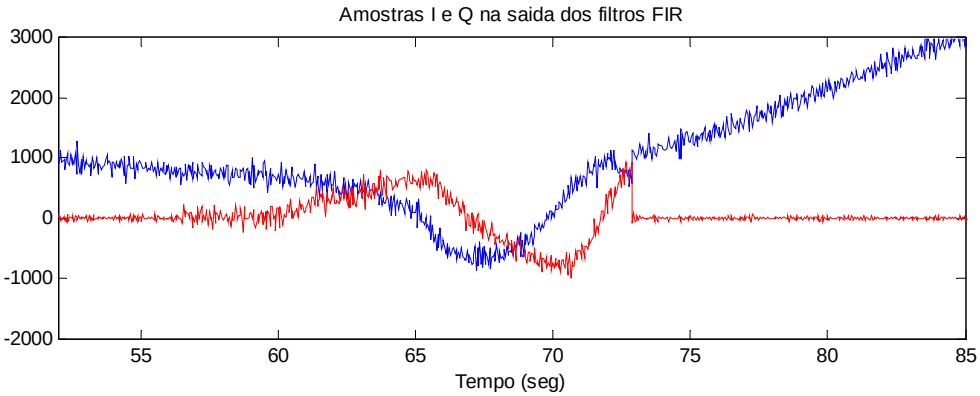

Figura 7.4 – Componentes I e Q em detalhe durante um "congelamento" do NCO.

Na Figura 7.4 estão apresentadas em detalhe as compoentes I e Q durante um "congelamento" do NCO. Estas componentes apresentam um comportamento sinusoidal característico a malha aberta e cuja frequência corresponde à diferença entre a frequência do sinal de entrada e a frequência a que o NCO é "congelado". Como se pode observar, estas sinusoides apresentam uma frequência bastante pequena (cerca de 0.09Hz), o que permite concluir a validade da solução encontrada para estimar a frequência a que deve ser "congelado" o NCO. Como o valor estimado é bastante próximo da frequência do sinal de entrada, a malha readquire o sincronismo quase instantaneamente.

Repetiu-se o teste anterior com uma frequência modulante de 0.03Hz e com um atenuador de 20dB entre o gerador de sinal e o *power splitter*. Os resultados estão representados na Figura 7.5.

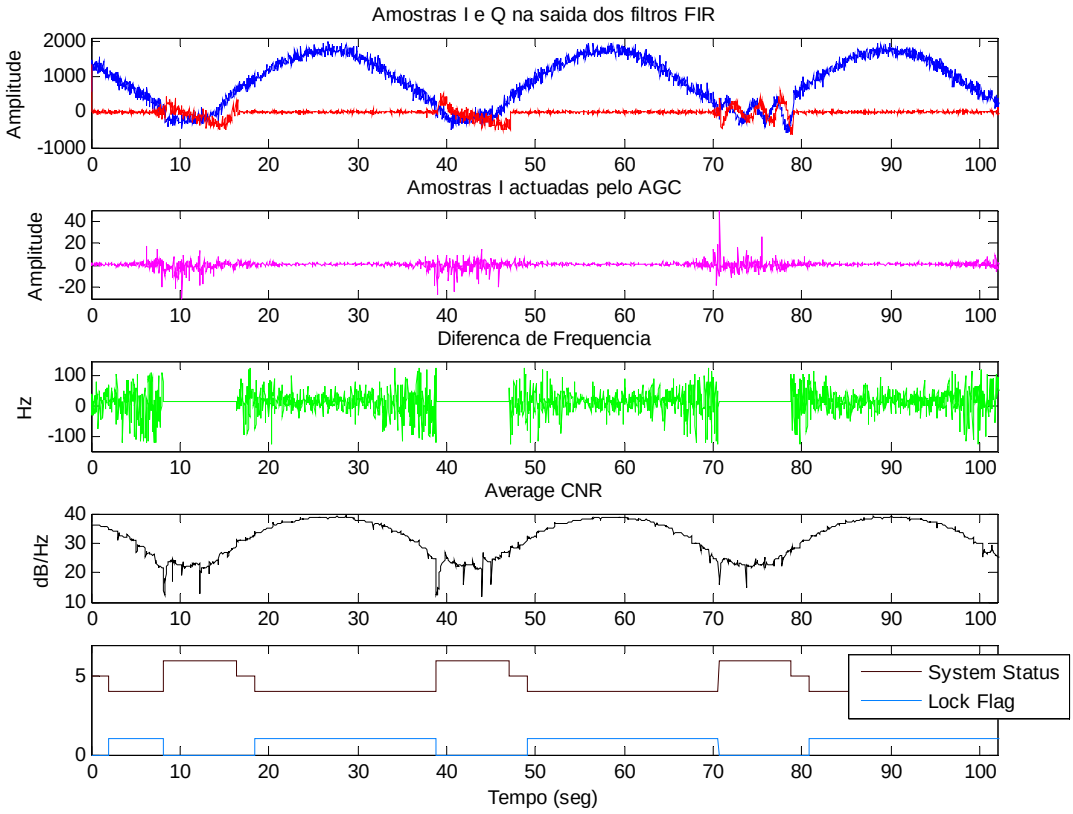

Figura 7.5 – Resultados obtidos com um atenuador de 20dB e a "congelar" o NCO abaixo de 27dB/Hz.

Como se pode observar, os resultados continuam a ser bastante satisfatórios quando se aumentou a atenuação do sinal em 10dB. O NCO é sempre "congelado" com sucesso mesmo com valores de CNR ainda mais reduzidos. Apesar da variância de frequência ser maior, a média de frequências usada para o "congelamento" do NCO continua a ser estimada com um valor bastante próximo da frequência do sinal de entrada.

Importa aqui lembrar o valor de desvio diário especificado para balizas de satélite (com valores medidos muito menores) e que foi referido anteriormente: 2kHz. Assumindo a variação de frequência como periódica, com periodicidade de 1 dia, teriamos uma taxa máxima de variação de frequência de 0.15Hz/s. Portanto um tempo de congelamento de algumas dezenas de segundos não parece inviabilizar deste modo a rápida reaquisição.

A estimativa de 5s para a frequência média do NCO é também uma forma de ter o melhor valor sempre actualizado. Um sistema destes realmente pode operar durante dezenas de dias sem haver situações que levem a PLL a perder o sincronismo.

7.2. Linearidade

Foram efectuados alguns testes para averiguar a linearidade do sistema, que consistiram na simples execução da PLL digital com o AGC activo durante

aproximadamente 30.7s (o equivalente a 150 000 amostras). Foram recolhidas as amostras da componente I e Q à saída da cadeia secundária de filtragem e decimação, assim como as amostras da componente I actuada pelo AGC e os sucessivos valores de frequência configurados no NCO. Efectuaram-se vários testes com diferentes amplitudes do sinal de entrada, com atenuações entre 0 e 24dB, com incrementos de 2dB e entre 24 e 30dB, com incrementos de 1dB. Analisando e efectuando alguns cálculos sobre as amostras recolhidas seria possível estimar a amplitude detectada pelo sistema. O objectivo destes testes seria verificar se a amplitude detectada correspondia à amplitude real do sinal de entrada. Os resultados obtidos foram analisados com o *m-file* presente em Anexos D.

Na Figura 7.6 está representado um gráfico da diferença entre a atenuação medida e a atenuação real (At_{out}-At_{in}) em função da atenuação real (At_{in}).

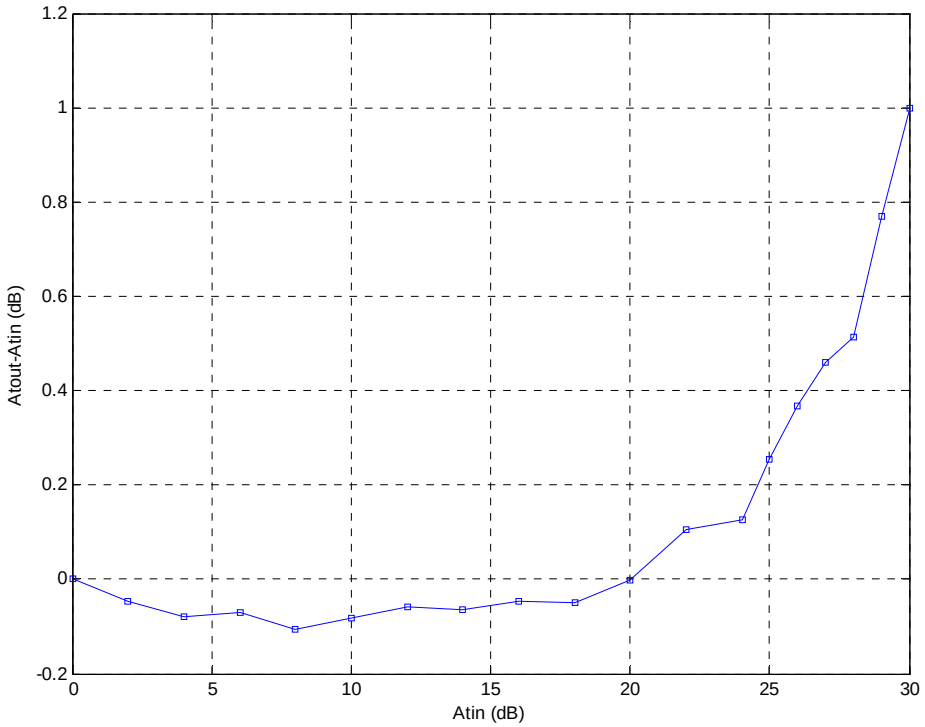

Figura 7.6 – Gráfico da diferença entre as atenuações medida e real em função da atenuação real.

Observando o gráfico anterior é possível constatar que a diferença entre a atenuação medida e a atenuação real é bastante pequena e não ultrapassa 0.1dB. No entanto, quando a atenuação real ultrapassa 25dB, a diferença em relação à atenuação medida aumenta significativamente, chegando a 1dB para uma atenuação real de 30dB. Estes resultados permitem concluir que o sistema se comporta linearmente até atenuações de 25dB e que acima deste valor o mesmo se verifica.

Na Figura 7.7 está representado um gráfico da variância de fase em função da relação sinal-ruído.

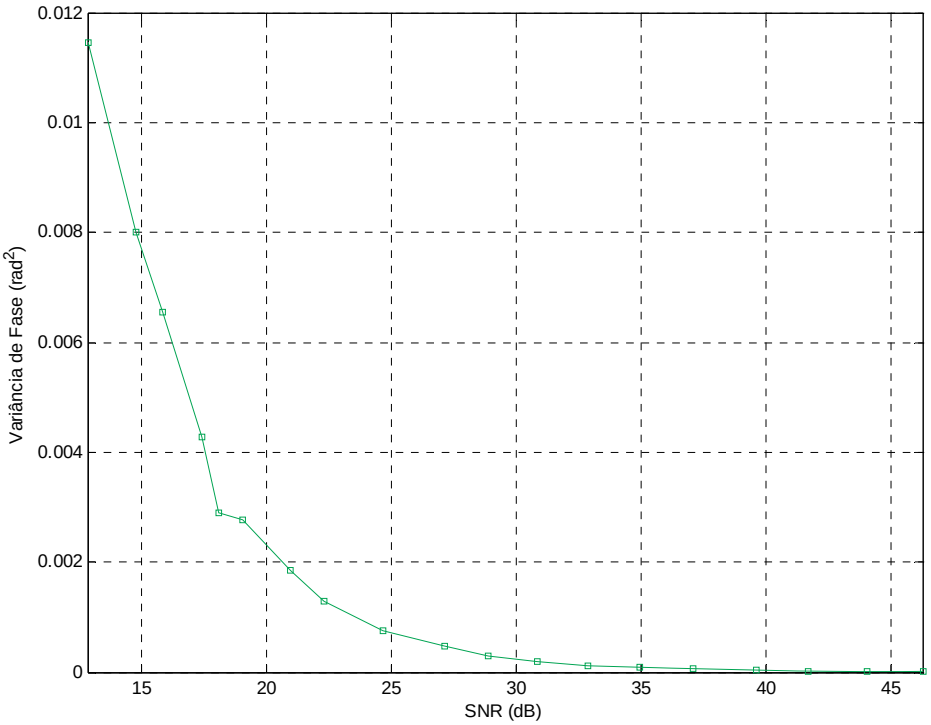

Figura 7.7 – Gráfico da variância de fase em função da relação sinal-ruído.

Como se pode observar no gráfico acima, a variância de fase à saída aumenta quando a SNR aumenta, ou seja, quando a amplitude do sinal diminui. No entanto, a variância de fase só aumenta consideravelmente quando a SNR é inferior a 25dB. Acima deste valor, a variância de fase mantém-se abaixo dos $0.001 rad^2$, sendo que neste caso a PLL consegue manter o sincronismo sem grandes problemas.

7.3. Outras Inspecções aos Resultados

Fizeram-se diversos outros testes que deram resultados similares aos previstos teóricamente.

Um teste que teria implicações na governação da malha foi o de avaliar o *offset* das componentes em fase e quadratura na ausência de sincronismo e com ruído apenas. Seria esperado obter um valor com média nula para ambas mas curiosamente nenhuma delas se parece aproximar deste valor. A componente Q apresenta um valor médio de 6 e a componente I um valor médio de 8, contra valores bem menores com carga adaptada. Portanto uma atenuação significativa do sinal combinada com qualquer dificuldade em detectar a perda de sincronismo conduz a que o NCO seja desviado rápidamente pela integração deste residual. Eventualmente a subtracção deste valor médio poderia minorar o problema mas o *software* desenvolvido não está dependente nem condicionado pelo facto descrito.

7.4. Resultados Obtidos com a FLL Digital

Foi efectuado um teste com a versão do programa com a FLL digital em que se utilizou um sinal de entrada sem ruído e sem modulação. Os resultados estão representados na Figura 7.8.

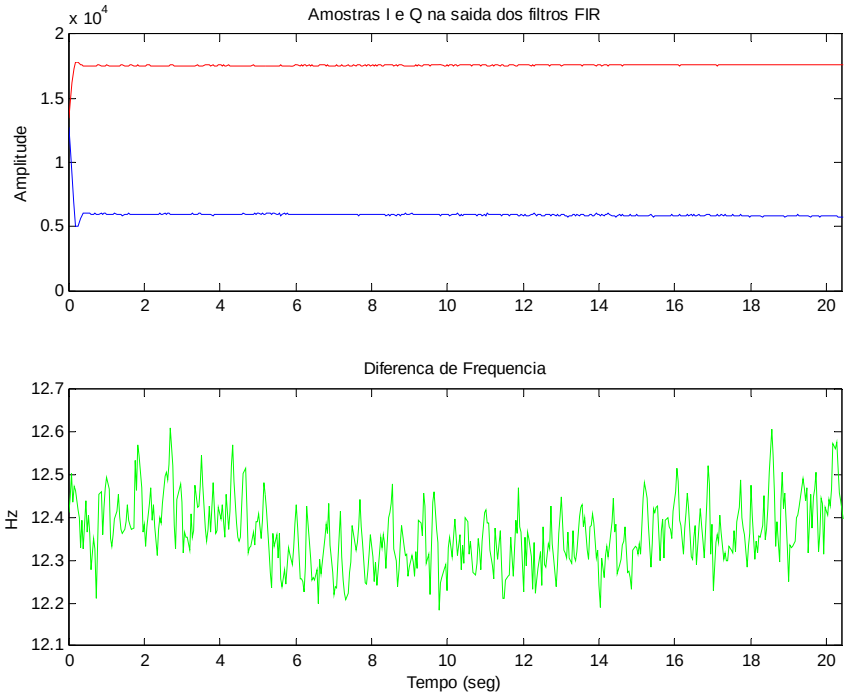

Figura 7.8 – Resultados obtidos com a FLL digital sem ruído e sem modulação.

Como se pode observar, na ausência de ruído a malha adquiriu o sincronismo quase instantaneamente e a variância de frequência é muito pequena, sendo que o NCO mantém uma frequência muito próxima do valor médio.

Repetiu-se o teste anterior com o sinal de entrada modulado e os resultados obtidos estão representados na Figura 7.9.

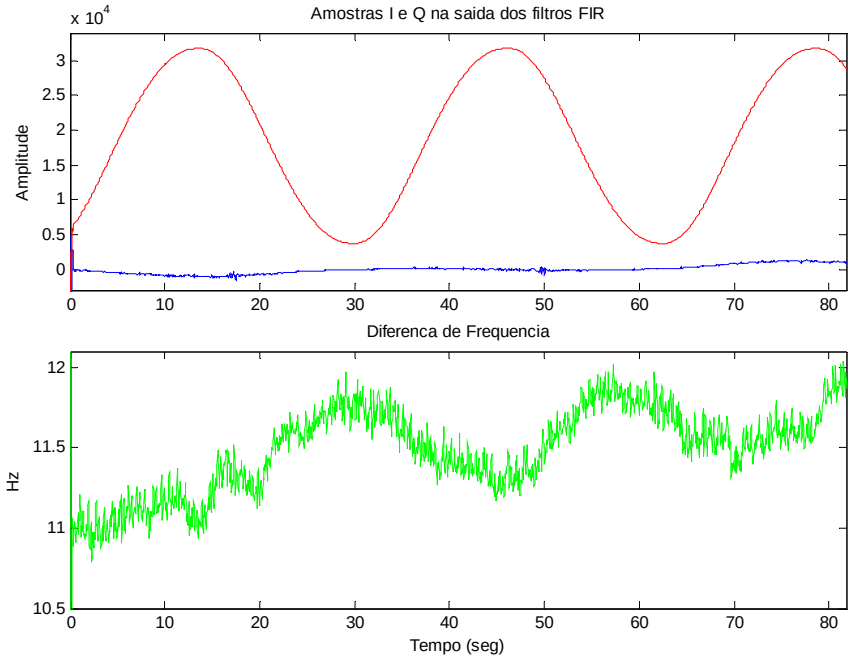

Figura 7.9 – Resultados obtidos com a FLL digital sem ruído e com modulação.

Repetiu-se o primeiro teste mas agora com o sinal de entrada adicionado de ruído.

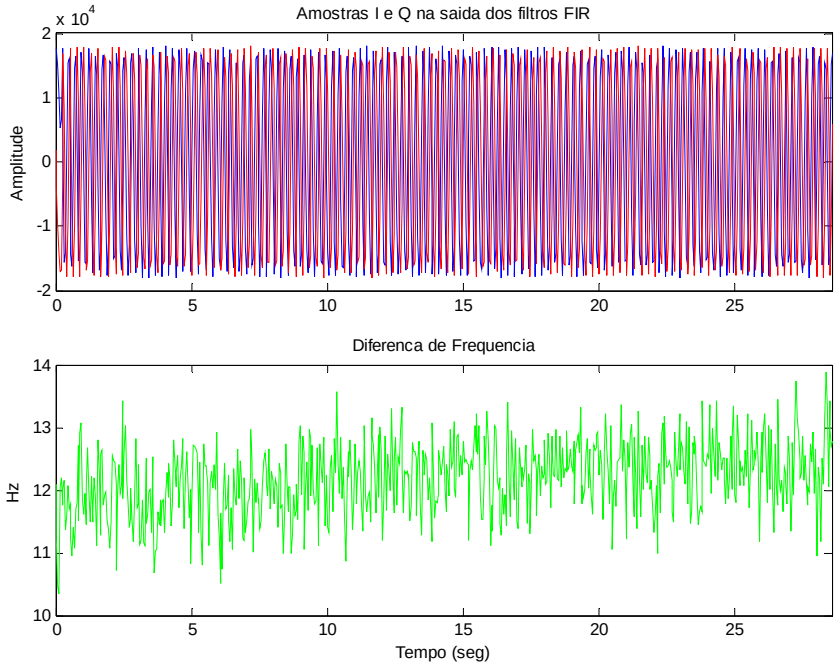

Figura 7.10 – Resultados obtidos com a FLL digital com ruído e sem modulação.

Como se pode observar na Figura 7.10, na presença de ruído, embora possa não parecer na primeira inspecção dos dados, a malha conseguiu adquirir o sincronismo já que a frequência não varia de forma significativa durante o longo período observado.

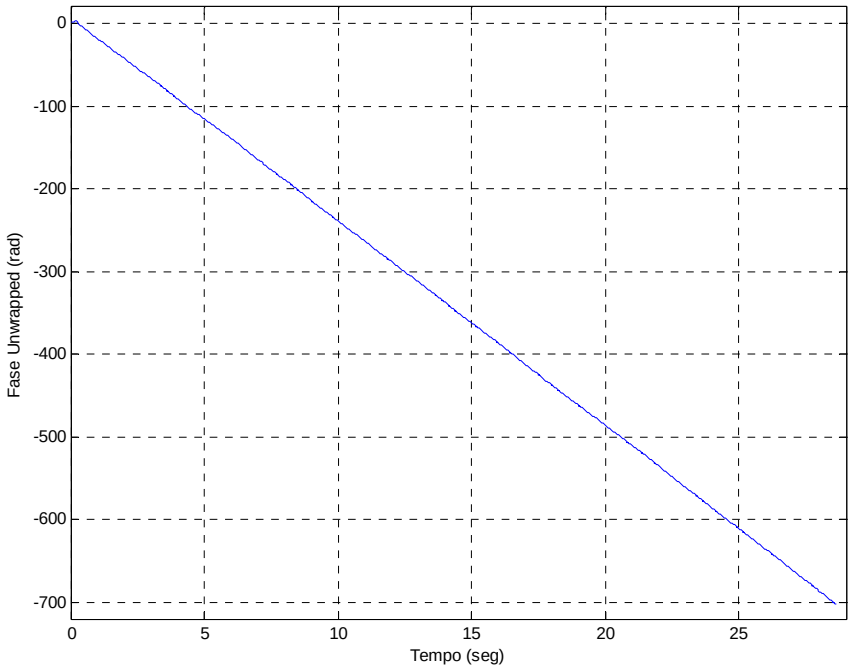

Figura 7.11 – Desvio de fase entre o sinal de entrada e o NCO.

Como se pode observar na Figura 7.11, no caso anterior há um acumular de fase de cerca de 702.5rad após cerca de 28s, o que implica um ligeiro desvio de frequência de aproximadamente 3.9Hz. A malha adquire o sincronismo mas com um erro de frequência que esta não é capaz de corrigir, eventualmente devido a um *offset* residual do detector de frequência.

Verificou-se ainda que ao contrário do que acontecia com a PLL, para atenuações de cerca de 20dB a malha não consegue adquirir o sincronismo.

A FLL não foi profundamente testada. O desempenho depende bastante da largura de banda do filtro que precede o detector de frequência e seguramente da largura de banda da malha. Urge aqui algum trabalho de maior profundidade sobre o desempenho desta malha, que leve igualmente em conta a influência do AGC que neste momento tem uma largura de banda da ordem da largura de banda da FLL.

7.5. Resultados Obtidos com o Protótipo de Dois Canais

Utilizando o protótipo de dois canais, repetiu-se um dos testes anteriores com um atenuador de 20dB, sendo que os resultados obtidos estão apresentados na Figura 7.12.

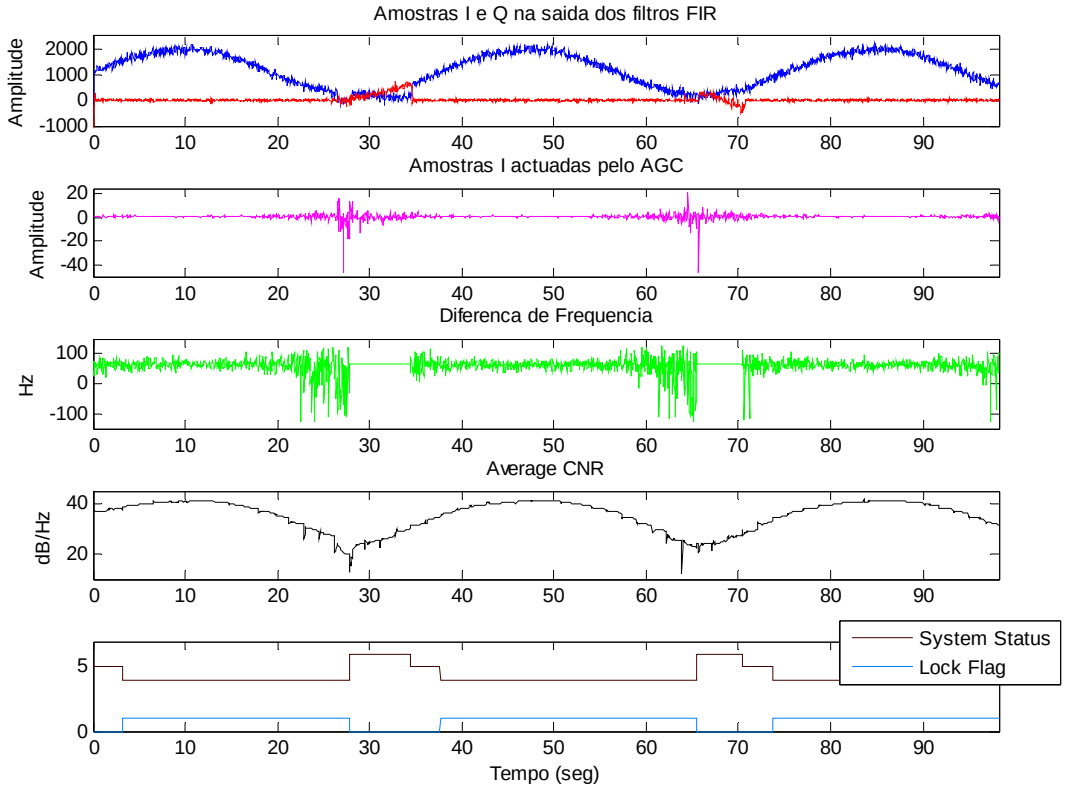

Figura 7.12 – Resultados obtidos com um atenuador de 20dB e a "congelar" o NCO abaixo de 27dB/Hz (protótipo de dois canais).

Comparando os resultados da figura acima com os resultados da Figura 7.5, pode-se concluir que o *software* desenvolvido para o protótipo de um canal foi migrado com sucesso para o protótipo de dois canais.

Uma vez que o teste anterior incluía apenas o sinal copolar (apesar do sinal crosspolar ser também processado), efectuou-se ainda um novo teste recorrendo a um T coaxial, para analisar simultaneamente os dois sinais recebidos. Este teste consistiu em recolher 300 000 amostras correspondentes aos canais copolar e crosspolar, com o sinal de entrada modulado e adicionado de ruído. Utilizou-se ainda um atenuador de 10dB intercalado entre a saída do sinal crosspolar no T coaxial.

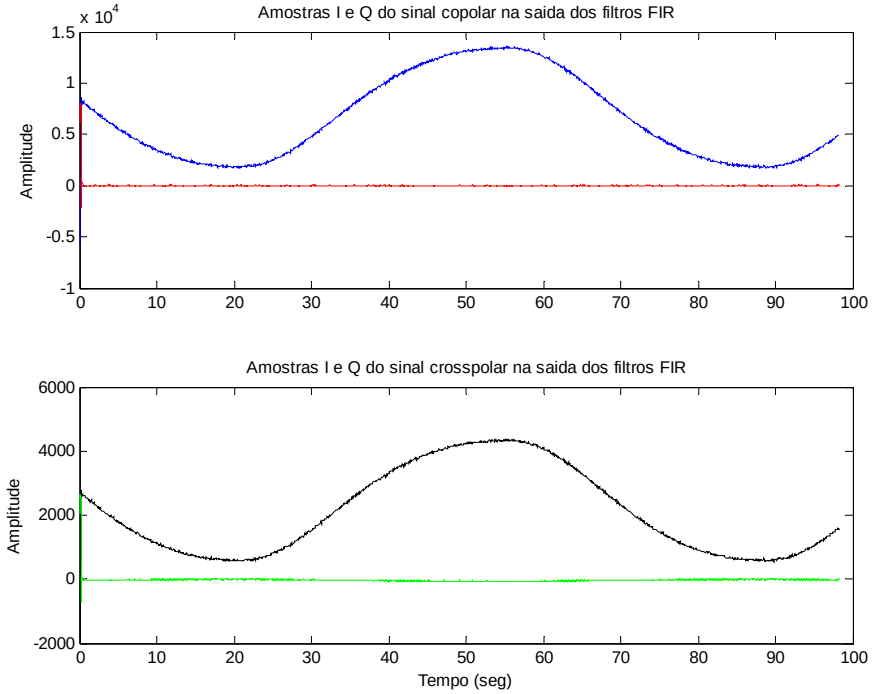

Figura 7.13 – Amostras I e Q dos sinais copolar e crosspolar.

Posteriormente repetiu-se o teste anterior mas diminuindo a amplitude do sinal em 10dB, utilizando um atenuador entre o gerador de sinal e o *power splitter*.

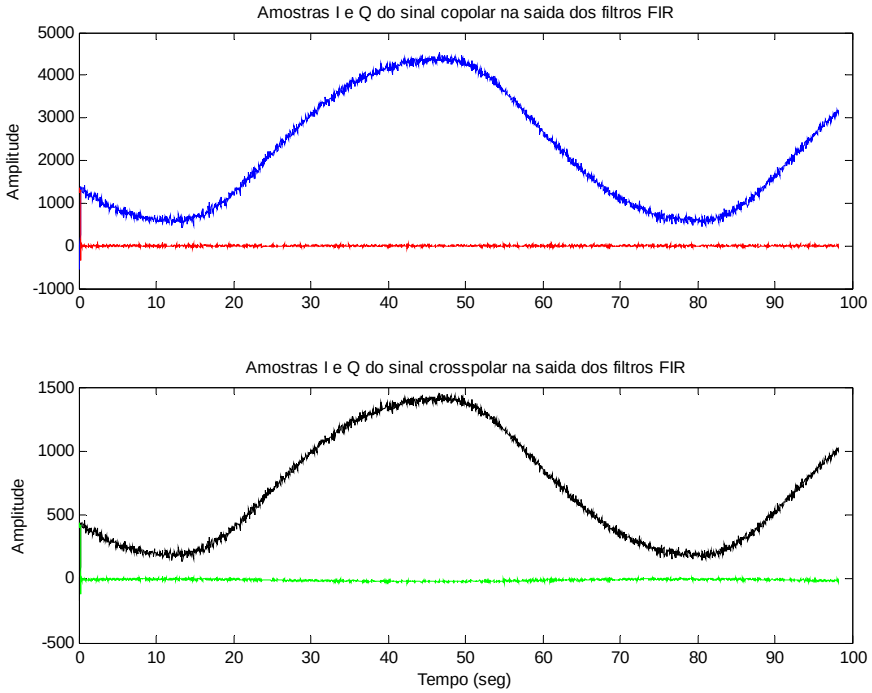

Figura 7.14 – Amostras I e Q dos sinais copolar e crosspolar com um atenuador de 10dB.

Na Figura 7.15, estão representados os resultados de uma repetição do teste anterior mas com um atenuador de 20dB.

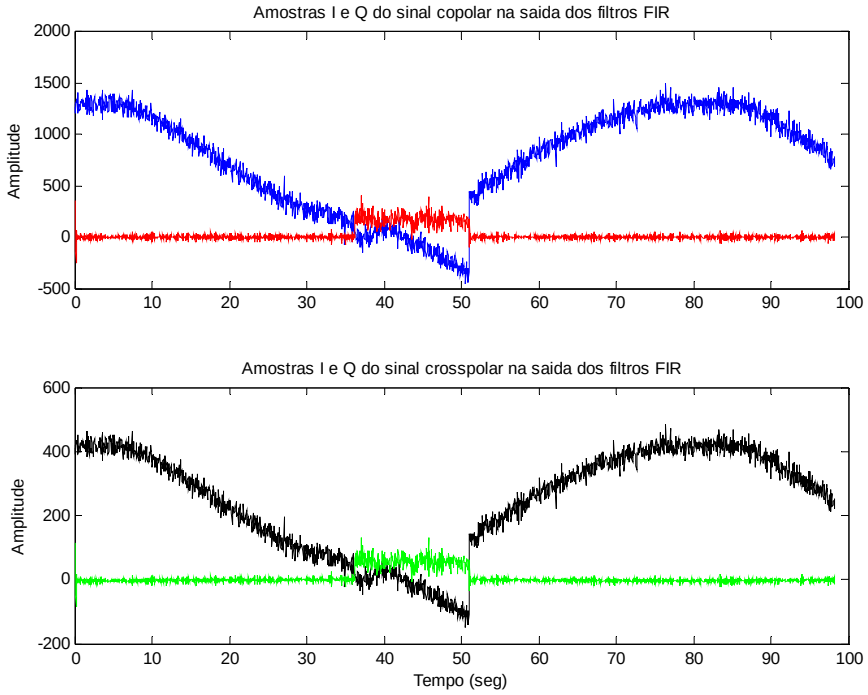

Figura 7.15 – Amostras I e Q dos sinais copolar e crosspolar com um atenuador de 20dB.

Como se pode observar nas figuras anteriores, ambos os canais copolar e crosspolar estão a ser correctamente processados.

Por fim repetiu-se o primeiro teste mas utilizando um cabo coaxial mais longo (com cerca de 80cm) no canal crosspolar, sendo que os resultados estão apresentados na Figura 7.16. Este teste pretende introduzir um diferencial de fase entre os dois canais, que é agora bem visível nesta última figura ao observar a componente Q.

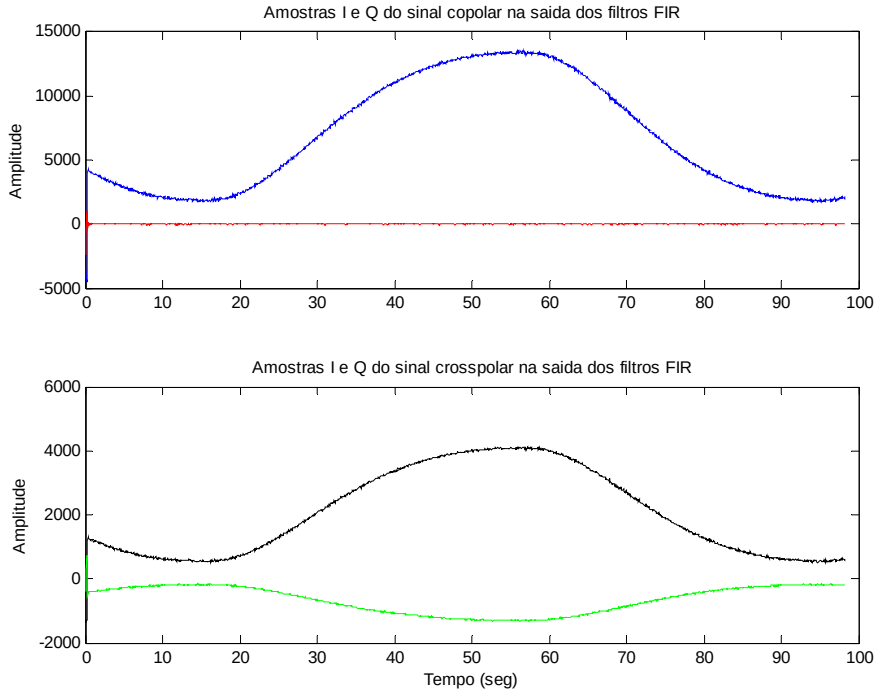

Figura 7.16 – Amostras I e Q dos sinais copolar e crosspolar com um cabo coaxial longo no canal crosspolar.

Na Figura 7.17 está representada a fase relativa entre o sinal copolar e o sinal crosspolar, onde se pode observar mais em concreto a diferença de fase introduzida pela utilização de um cabo mais longo e para uma amplitude de variação do sinal de sinal similar ao caso anterior. A fase relativa apresenta uma variação de ±0.4° em torno do valor médio de 17.85°, o que é um valor perfeitamente aceitável para este tipo de medida. Atendendo ao comprimento do cabo (cerca de 80cm) e à respectiva constante dieléctrica, o valor da fase relativa entre os dois canais é muito semelhante ao esperado.

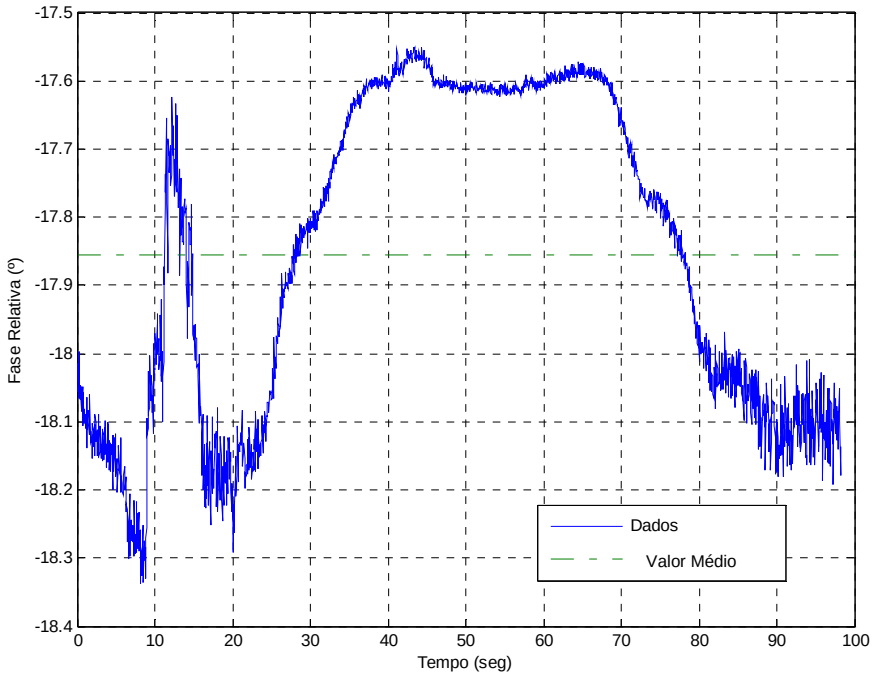

Figura 7.17 – Fase relativa entre o sinal copolar e o sinal crosspolar.

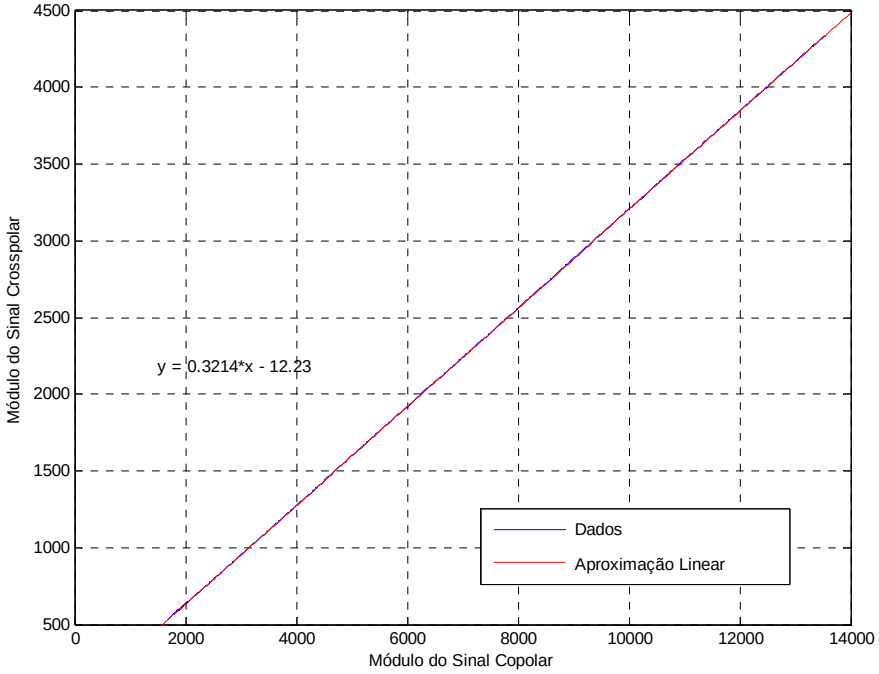

Figura 7.18 – *Scatter plot* das componentes I dos sinais copolar e crosspolar.

Na Figura 7.18 está representado um *scatter plot* com a variação do módulo do sinal crosspolar em função do módulo do sinal copolar, sendo que o comportamento observado é linear e os dados sobrepõem-se praticamente à recta obtida por linearização. Convertendo o valor do declive desta recta para dB, obtém-se aproximadamente -9.9dB, o que como seria de esperar, corresponde aproximadamente ao valor do atenuador de 10dB intercalado entre o T e a saída do canal crosspolar.

8. Conclusões e Trabalho Futuro

Com a conclusão deste trabalho, é possível afirmar que os objectivos propostos foram globalmente atingidos:

- Foi implementada uma solução eficiente para a estimação da frequência inicial do sinal de entrada no arranque do sistema;
- O AGC foi implementado com sucesso e permite manter os parâmetros da malha constantes independentemente da atenuação do sinal;
- Foi desenvolvido um algoritmo de administração do sistema bastante eficiente, que avalia as condições de funcionamento da malha e administra-a consoante as necessidades;
- A cadeia secundária de filtragem e decimação permite a disponibilização das componentes em fase e quadratura a uma taxa de 24.41S/s, que poderão futuramente ser enviadas para o PC anfitreão e arquivadas em ficheiro.

Este trabalho permitiu a aquisição de novos conhecimentos em várias áreas e também a solidificação dos conhecimentos adquiridos no projecto final de 5º ano. De entre as diversas áreas abrangidas neste trabalho, destacam-se as seguintes:

- Aprofundamento da familiarização com arquitecturas de processadores cada vez mais utilizados em comunicações móveis e outras aplicações: as DSPs;
- Programação em linguagem C dos diversos módulos de *software* necessários para o sistema;
- Utilização de algumas ferramentas de trabalho poderosas, nomeadamente o *Code Composer Studio*, utilizado para o desenvolvimento de *software* para o sistema e o *Matlab*, a já conhecida ferramenta de cálculo;
- Processos de *debugging* para resolução de problemas de *software*;
- Manuseamento de equipamentos essenciais para testes de sistemas electrónicos;
- Aplicações de processamento digital de sinal e conceitos de controlo linear;
- Aprofundamento do estudo das malhas de captura de fase ou PLL e implementação de uma PLL digital;
- Estudo da malha de captura de frequência ou FLL;
- Estudo e implementação de um sistema de controlo automático de ganho ou AGC.

O método desenvolvido para estimar a frequência inicial do sinal de entrada no arranque do sistema, revelou-se muito eficaz mesmo para sinais adicionados de ruído e com atenuações elevadas. No entanto, existem algumas limitações que devem ser futuramente corrigidas e que se prendem com a função utilizada para o cálculo das FFTs, que exige demasiada memória. Para melhorar a estimativa da densidade espectral

de potência de ruído seria necessário utilizar um maior número de pontos para o cálculo das FFTs.

Relativamente ao AGC, pode-se concluir que o modelo implementado apresenta um desempenho bastante bom com variações elevadas da amplitude do sinal e mesmo na presença de ruído. Em todos os testes efectuados, a componente I actuada pelo AGC mantinha sempre uma média muito próxima de 1 e raramente inferior a 0.9 de acordo com o previsto teóricamente. Apesar da variância desta componente aumentar muito com a atenuação do sinal, a malha conseguia sempre manter o sincronismo sem ser necessário alterar-lhe nenhum parâmetro. Assim, de acordo com os objectivos propostos, foi possível a utilização de valores constantes para os parâmetros da malha mesmo com a amplitude do sinal de entrada a variar ao longo do tempo.

O algoritmo de administração da malha foi implementado com sucesso, permitindo gerir o funcionamento do sistema de acordo com algumas condições observadas ao longo do tempo. Estas condições são avaliadas através da indicação de sincronismo e da estimação do valor de CNR do sinal de entrada. Quando estas condições revelam uma situação mais provável de perda de sincronismo, o sistema toma algumas medidas para que tal se possa evitar, procedendo ao "congelamento" do NCO. Esta solução revelou-se bastante eficiente uma vez que ao abrir a malha e manter o NCO a uma frequência constante, é possível a reaquirir posteriormente o sincronismo desde que a frequência a que o NCO é "congelado" não se afaste muito da frequência do sinal de entrada. Uma vez que não se esperam variações rápidas de frequência no sinal de entrada, ao "congelar" o NCO a uma frequência correspondente ao último valor médio de frequência obtido antes de abrir a malha e com um bom valor de CNR, garante-se que a probabilidade da malha readquirir o sincronismo é elevada.

De seguida, apresentam-se algumas sugestões de trabalho futuro:

- Relativamente à transferência dos dados para o PC, por falta de tempo não foi possível testar a solução descrita em [8]. No entanto, como se referiu na secção 3.6, esta solução é implementada em grande parte por *software*, pelo que poderá ser introduzido um *overhead* indesejável para o funcionamento do sistema. Surgem então duas soluções alternativas.

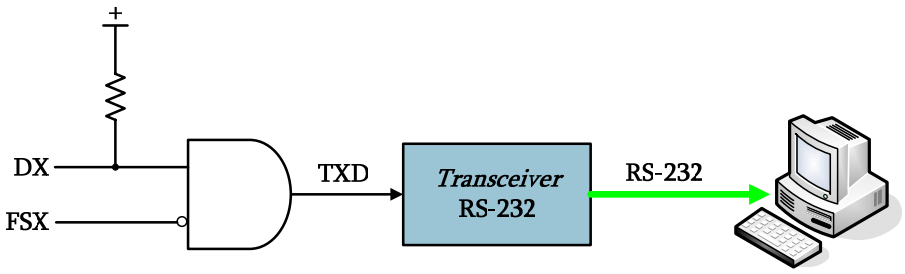

Figura 8.1 – Solução alternativa para a emulação de uma transferência de dados RS-232.

A solução mais simples está representada na Figura 8.1 e baseia-se na utilização de uma *gate* lógica, uma resistência de *pull-up* e de um *transceiver* RS-232. Todavia, esta solução só será viável se for possível programar o período do relógio da porta série do porto McBSP1 (CLKX), com um dos valores padrão do protocolo RS-232. Para considerar esta solução válida é também necessário que o diagrama observado em [16, pág. 20] corresponda exactamente à realidade. Por outras palavras, a linha de transmissão de dados DX tem de estar no estado de alta impedância quando não estão a ser transmitidos bits e o impulso de sincronização FSX tem que ocorrer antes do início da transmissão do primeiro bit e prolongar-se durante um período de transmissão de um bit. Desta forma, o impulso FSX permitiria gerar um *start bit* antes do envio de cada *frame*. Um outro requisito no que toca à geração do *stop bit*, consiste na necessidade de introduzir uma pausa entre a transmissão de *frames* consecutivas.

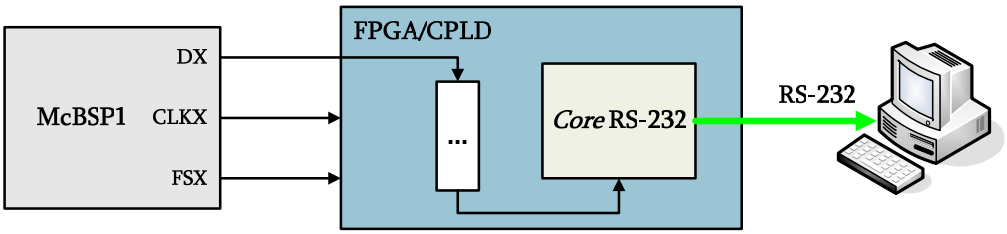

Figura 8.2 – Solução alternativa baseada na utilização de um dispositivo FPGA/CPLD.

Na Figura 8.2 está representada uma outra solução para a emulação de uma transferência de dados com o protocolo RS-232, que se baseia na utilização de um dispositivo programável adicional, que poderá ser um CPLD (*Complex Programmable Logic Device*) ou uma FPGA (*Field-Programmable Gate Array*). Enquanto o sistema está em execução, os dados seriam enviados para um *buffer* presente neste dispositivo programável. Quando o *buffer* fica cheio, os dados são enviados para um *core* RS-232, que por sua vez implementa a transferência para o PC anfitreão de acordo este protocolo;

- Repetição de alguns testes com um ruído gerado de forma mais realista, para obter um diagnóstico mais aprofundado do sistema, quer relativamente à estimação da frequência do sinal de entrada como também ao algoritmo de administração da malha;
- Execução de novos testes para o aprofundamento da avaliação do comportamento da FLL digital;
- Substituição da rotina utilizada para o cálculo das FFTs, por uma das rotinas disponibilizadas pela biblioteca da DSP, que além de serem executadas mais rapidamente, não exigem tanta memória e logo permitiriam a utilização de resoluções espectrais mais elevadas.

Referências Bibliográficas

[1] A. P. DEkker, "The effect of Thermal Noise and Phase Noise on the accuracy of Amplitude and Phase Measurement," presented at DOC-OPEX 7.

[2] F. M. Gardner, *Phaselock Techniques*: John Wiley & Sons, Inc., 1966.

[3] R. Sousa and J. Pires, "Projecto de 5º Ano - Receptor Digital para Medição de Balizas de Satélite," Universidade de Aveiro, Aveiro 2006.

[4] F. D. Natali, "AFC Tracking Algorithms," in *IEEE Transations on communications*, 1984.

[5] A. Moldovan, "FFL," Universidade de Aveiro, Aveiro 2007.

[6] L. Cutler and C. Searl, "Some aspects of the theory and measurement of frequency fluctuations in frequency standards," *Preceedings IEEE*, vol. 54, pp. 136, 1966.

[7] J. M. Cargal, *Discrete Mathematics for Neophytes: Number Theory, Probability, Algorithms, and Other Stuf*.

[8] "TMS320C6000 McBSP - UART," Texas Instruments Incorporated, 2004.

[9] "TMS320C6000 Programmer's Guide," Texas Instruments Incorporated, 2006.

[10] "TMS320C6000 Optimizing Compiler v 6.0 Beta - User's Guide," Texas Instruments Incorporated, 2005.

[11] "TMS320C67x DSP Library - Programmer's Reference Guide," Texas Instruments Incorporated, 2006.

[12] D. Cross, "FFT C source code (Simple radix-2)," www.yov408.com, 2005.

[13] J. Pires, "Receptor de Dois Canais para Balizas de Satélite," Universidade de Aveiro, Aveiro 2007.

[14] "AD9850 - CMOS, 125MHz Complete DDS Synthesizer," Analog Devices Incorporated, 2004.

[15] "TMS320C6000 DSP External Memory Interface (EMIF) Reference Guide," Texas Instruments Incorporated, 2005.

[16] "TMS320C6000 DSP Multichannel Buffered Serial Port (McBSP) Reference Guide," Texas Instruments Incorporated, 2005.

[17] "AD6620 – 67 MSPS Digital Receive Signal Processor." Norwood, M.A.:.Analog Devices Incorporated, 2001.

Anexos

A. Estimação da Frequência do Sinal de Entrada

```
%Leitura das amostras recolhidas
amostras1 = load('amostras1.txt');  %Dif Freq = +50Hz;
amostras2 = load('amostras2.txt');  %Dif Freq = -50Hz;
amostras3 = load('amostras3.txt');  %Dif Freq = +500Hz;
amostras4 = load('amostras4.txt');  %Dif Freq = -500Hz;
amostras5 = load('amostras5.txt');  %Dif Freq = +750Hz;
amostras6 = load('amostras6.txt');  %Dif Freq = -750Hz;
amostras7 = load('amostras7.txt');  %Dif Freq = +1000Hz;
amostras8 = load('amostras8.txt');  %Dif Freq = -1000Hz;

% Os arrays com as partes reais e imaginarias foram imprimidos nos
% ficheiros "amostra<x>.txt" da seguinte forma:
      % realIn<x> = amostras<x>(1:end,1); -> Coluna 1
      % imagIn<x> = amostras<x>(1:end,2); -> Coluna 2
      % realOut<x> = amostras<x>(1:end,3); -> Coluna 3
      % imagOut<x> = amostras<x>(1:end,4); -> Coluna 4

N = length(amostras1(1:end,1));   % Numero de amostras para o calculo da FFT

% Calculo do modulo das amostras do espectro da FFT complexa obtido na DSP
% absFFT<x> = sqrt(realOut<x>^2 + imagOut<x>^2)/N
absFFT1 = sqrt(amostras1(1:end,3).^2 + amostras1(1:end,4).^2)/N;
absFFT2 = sqrt(amostras2(1:end,3).^2 + amostras2(1:end,4).^2)/N;
absFFT3 = sqrt(amostras3(1:end,3).^2 + amostras3(1:end,4).^2)/N;
absFFT4 = sqrt(amostras4(1:end,3).^2 + amostras4(1:end,4).^2)/N;
absFFT5 = sqrt(amostras5(1:end,3).^2 + amostras5(1:end,4).^2)/N;
absFFT6 = sqrt(amostras6(1:end,3).^2 + amostras6(1:end,4).^2)/N;
absFFT7 = sqrt(amostras7(1:end,3).^2 + amostras7(1:end,4).^2)/N;
absFFT8 = sqrt(amostras8(1:end,3).^2 + amostras8(1:end,4).^2)/N;

% Calculo do modulo das amostras do espectro da FFT complexa obtido no Matlab
% Este calculo e feito com as mesmas amostras de entrada utilizadas para o
% calculo do espectro na DSP
% absFFTideal<x> = sqrt(realIn<x>^2 + imagIn<x>^2)/N
absFFTideal1 = abs(fft(amostras1(1:end,1) + j*amostras1(1:end,2)))/N;
absFFTideal2 = abs(fft(amostras2(1:end,1) + j*amostras2(1:end,2)))/N;
absFFTideal3 = abs(fft(amostras3(1:end,1) + j*amostras3(1:end,2)))/N;
absFFTideal4 = abs(fft(amostras4(1:end,1) + j*amostras4(1:end,2)))/N;
absFFTideal5 = abs(fft(amostras5(1:end,1) + j*amostras5(1:end,2)))/N;
absFFTideal6 = abs(fft(amostras6(1:end,1) + j*amostras6(1:end,2)))/N;
absFFTideal7 = abs(fft(amostras7(1:end,1) + j*amostras7(1:end,2)))/N;
absFFTideal8 = abs(fft(amostras8(1:end,1) + j*amostras8(1:end,2)))/N;

% Graficos com os espectros obtidos na DSP e no Matlab
subplot(2,2,1),plot(absFFTideal1),hold on, plot(absFFT1,'r'), hold on, plot(absFFTideal2,'k'),hold on,
plot(absFFT2,'g'), title('Dif Freq = +/-50Hz'), xlabel('N'), ylabel('Amplitude'), legend('FFT no Matlab (Dif
Freq Positivas)','FFT na DSP (Dif Freq Positivas)', 'FFT no Matlab (Dif Freq Negativas)','FFT na DSP (Dif
Freq Negativas)'), AXIS([1 512 0 18000]);
```

```
subplot(2,2,2),plot(absFFTideal3),hold on, plot(absFFT3,'r'), hold on, plot(absFFTideal4,'k'),hold on,
plot(absFFT4,'g'), title('Dif Freq = +/-500Hz'), xlabel('N'), ylabel('Amplitude'), AXIS([1 512 0 18000]);
subplot(2,2,3),plot(absFFTideal5),hold on, plot(absFFT5,'r'), hold on, plot(absFFTideal6,'k'),hold on,
plot(absFFT6,'g'), title('Dif Freq = +/-750Hz'), xlabel('N'), ylabel('Amplitude'), AXIS([1 512 0 18000]);
subplot(2,2,4),plot(absFFTideal7),hold on, plot(absFFT7,'r'), hold on, plot(absFFTideal8,'k'),hold on,
plot(absFFT8,'g'), title('Dif Freq = +/-1000Hz'), xlabel('N'), ylabel('Amplitude'), AXIS([1 512 0 18000]);
```

B. Cadeia Secundária de Filtragem e Decimação

```
Freq_CLK = 40*10^6; %frequencia do sinal de relogio do AD6620 - 40MHz
MCIC2 = 8; %factor decimador do filtro CIC2 do AD6620
MCIC5 = 32; %factor decimador do filtro CIC5 do AD6620
MRCF = 32; %factor decimador do filtro RCF do AD6620
fsamp5 = Freq_CLK/(MCIC2*MCIC5); %frequencia de amostragem na entrada do filtro RCF
fsampr = fsamp5/MRCF; %frequencia de amostragem na saida do filtro RCF

M1 = 40;    %factor de decimacao do primeiro filtro FIR
M2 = 5;     %factor de decimacao do segundo filtro FIR
N1 = 3*M1;  %numero de amostras do primeiro buffer
N2 = 10*M2;   %numero de amostras do segundo buffer

freqSamp = fsampr;    %frequencia de amostragem do sinal na entrada do primeiro filtro FIR
freqSamp2 = freqSamp/M1;    %frequencia de amostragem do sinal na entrada do segundo filtro FIR

T = 1/freqSamp; %periodo de amostragem
N = 20000;  %numero de amostras
Nextra = M1*(N2-M2+N1/M1-1);   %numero de amostras extra
freq = 2;   %frequencia do sinal de entrada
t = 0:T:(N+Nextra-1)*T;    %instantes de amostragem do sinal de entrada
Amp = 20000;    %amplitude do sinal de entrada
x = Amp*sin(2*pi*freq*t);  %sinal de entrada

fc = 30; %frequencia de corte do primeiro filtro FIR
fa = 100; %frequencia de atenuacao do primeiro filtro FIR (no maximo 109Hz)
rc = 0.01; %ripple na passband
ra = 0.025; %ripple na stopband
%Determinacao dos coeficientes do primeiro filtro FIR
[n,fo,mo,w] = firpmord( [fc fa], [1 0], [rc ra], freqSamp );
h1 = firpm(n,fo,mo,w);

fc2 = freqSamp2/(2*M2); %frequencia de corte do segundo filtro FIR
fa2 = 18; %frequencia de atenuacao do segundo filtro FIR
rc2 = 0.005; %ripple na passband do segundo filtro FIR
ra2 = 0.005; %ripple na stopband do segundo filtro FIR
%Determinacao dos coeficientes do segundo filtro FIR
[n2,fo2,mo2,w2] = firpmord( [fc2 fa2], [1 0], [rc2 ra2], freqSamp2);
h2 = firpm(n2,fo2,mo2,w2);

%normalizacao dos coeficientes a 1
h1 = h1/sum(h1);
h2 = h2/sum(h2);

%Grafico da resposta em frequencia do primeiro filtro FIR
figure(1);
```

```
freqz(h1);
title('Resposta em frequencia do filtro FIR1');

%Grafico da resposta em frequencia do segundo filtro FIR
figure(2);
freqz(h2);
title('Resposta em frequencia do filtro FIR2');

%aplicacao do primeiro filtro FIR
for k = 0:((N+Nextra-(N1/M1-1)*M1)/M1-1)
   y = 0;
   for i = 1:N1
      y = y+h1(i)*x(i+k*M1);
   end
   y1(k+1) = y;
end

%aplicacao do segundo filtro FIR
for k = 0:((N+Nextra-(N1/M1-1)*M1-(N2/M2-1)*M1*M2)/(M1*M2)-1)
   y = 0;
   for i = 1:N2
      y = y+h2(i)*y1(i+k*M2);
   end
   y2(k+1) = y;
end

t1 = 0:T:(N+Nextra-(N1/M1-1)*M1-1)*T;   %intantes de amostragem na entrada
t2 = ((N1/M1-1)*M1+(N2/M2-1)*M1*M2)*T:M1*M2*T:(N+Nextra-1)*T;   %instantes de amostragem na
saida dos filtros

figure(3);
plot(t1,x(1:N+Nextra-2*M1)), hold on, plot(t2,y2,'r'), grid on, xlabel('Tempo (seg)'), ylabel('Amplitude'),
legend('sinal de entrada','sinal filtrado e decimado');    %graficos anteriores sobrepostos

%Impressão dos coeficientes do filtro FIR1
for i = 1:length(h1)
fprintf('%5.8f,\n', h1(i));
end

%Impressão dos coeficientes do filtro FIR2
for i = 1:length(h2)
fprintf('%5.8f,\n', h2(i));
end
```

C. PLL Digital

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definicao de constantes
%%%%%%%%%%%%%%%%%%%%%%%%%%%
qsi = 0.707; %coeficiente de amortecimento
BL = 50; %largura de banda
amplitude_maxima = 18500;
freq_amostragem = 40*10^6; %frequencia de amostragem
MCIC2 = 8; %factor decimador do filtro CIC2 do AD6620
```

```
MCIC5 = 32; %factor decimador do filtro CIC5 do AD6620
MRCF = 32; %factor decimador do filtro RCF do AD6620
fsamp = freq_amostragem/(MCIC2*MCIC5*MRCF); %frequencia de amostragem na saida do AD6620
T = 1/fsamp; %periodo de amostragem na saida do AD6620
%%%%%%%%%%%%%%%%%%%%%%%%%

kd = log2(amplitude_maxima); %ganho do detector de fase
k0 = freq_amostragem/2^32; %ganho do VCO
wn = BL*2/(qsi+1/(4*qsi)); %frequencia natural

%calculo das constantes de tempo tau1 e tau2
tau1 = k0*kd/wn^2;
tau2 = (2*qsi)/(sqrt(k0*kd/tau1));

%calculo das constantes do filtro digital
C1 = T^2/tau1;
C2 = T*tau2/tau1;

f = 1/2000: 1/2000: 0.5; % frequência normalizada
f1 = f*fsamp; % vector com as frequencias
s = 1i*2*pi*f1; % vector com os valores de z
z = exp(1i*2*pi*f); % vector com os valores de z

%resposta em frequencia H(s)
Hs = (k0*kd*(s.*tau2+1)./tau1)./(s.^2+s.*(k0*kd*tau2./tau1)+k0*kd./tau1);

%resposta em frequencia H(z)
Hz = (k0*kd*(C1./(1-z.^-1)+C2))./(1-z.^-1)./(1+(k0*kd*(C1./(1-z.^-1)+C2))./(1-z.^-1));

%grafico da resposta em frequencia H(s) e H(z)
%subplot(2,1,1),
semilogx(f,20*log10(abs(Hs))), hold on, semilogx(f,20*log10(abs(Hz)),'r'), axis([0, 0.5, -45, 10]), grid on,
xlabel('f/fsamp'), ylabel('H(s) (dB)');
```

D. Análise dos Resultados dos Testes de Linearidade

```
%%%%%%%%%%%%%%%%%%%%%%%%%
% Definicao de constantes
%%%%%%%%%%%%%%%%%%%%%%%%%
Freq_CLK = 40*10^6; %frequencia do sinal de relogio do AD6620 - 40MHz
MCIC2 = 8; %factor decimador do filtro CIC2 do AD6620
MCIC5 = 32; %factor decimador do filtro CIC5 do AD6620
MRCF = 32; %factor decimador do filtro RCF do AD6620
fsamp5 = Freq_CLK/(MCIC2*MCIC5); %frequencia de amostragem na entrada do filtro RCF
fsampr = fsamp5/MRCF; %frequencia de amostragem na saida do filtro RCF

M1 = 40;    %factor de decimacao do primeiro filtro FIR
M2 = 5;     %factor de decimacao do segundo filtro FIR
N1 = 3*M1;  %numero de amostras do primeiro buffer
N2 = 10*M2;   %numero de amostras do segundo buffer

freqSamp = fsampr;    %frequencia de amostragem do sinal na entrada do primeiro filtro FIR
freqSamp1 = freqSamp/M1;    %frequencia de amostragem do sinal na entrada do segundo filtro FIR
freqSamp2 = freqSamp1/M2;   %frequencia de amostragem do sinal na saida do segundo filtro FIR
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%

amostras = load('amostras.txt');

I = amostras(1:end,1);
Q = amostras(1:end,2);
I_AGC = amostras(1:end,3);
freqs = amostras(1:end,4);

T = 1/freqSamp; %periodo de amostragem
t = 0:T*M1*M2:(length(amostras)-1)*T*M1*M2;   %intantes de amostragem na entrada

subplot(3,1,1), plot(t,I), hold on, plot(t,Q,'r'), title('Amostras I e Q na saida dos filtros FIR');
subplot(3,1,2), plot(t,I_AGC,'m'), title('Amostras I actuadas pelo AGC');
subplot(3,1,3), plot(t,freqs-10700400,'g'), title('Diferenca de Frequencia');

% The programa analysis the results of detected signal in the card
%
% I and Q sampling rate is freqSamp2
%
% Aquisition data rate is freqSamp.
% Processes are seen as ergodic
%*********************************************************************

format long;

IMaxValue=1.7470e+004; % Maximum input value relative to which attenuation is calculated

I(1:10)=I(11:20); %Discard ...
Q(1:10)=Q(11:20); %... transiente due to aquisition

% Compute generaral data
IMean=mean(I) % Mean value of I and Q (Q mean must be close to 0)
QMean=mean(Q)
FreqMean=mean(freqs) %Mean of NCO frequency (assumed constant during the test)
FreqVar=var(freqs);
NCOPhaseStd=FreqVar .*(2.*pi/freqSamp)^2 %NCO phase variance (rad^2)

Atenua = 20.*log10(IMaxValue/IMean) %Computing attenuation

I_AGCMean=mean(I_AGC); %Mean Value de I_AGC
I_AGCVar=var(I_AGC); % Var I_AGC

%*********************************************************************
% Now compute estimated signal parameters from I time series
% Original time resolution is used here that is expected to be used for
% scintillation measurements
%
IFluct=I-IMean;
CNR=10 .*log10(IMean^2/(var(I-IMean)/14)) % Compute CNR from: Noise spectral density
                                          % and realtive Power/(Noise
                                          % spectral density). 14HZ is
                                          % the BW of LowpassFilter
IVardB=var(20 .*log10(I/IMean)) %Variance (dB^2 )

SNR = CNR-10*log10(14); % Computing SNR
```